NAMED ENTITY RECOGNITION FOR TURKISH MICROBLOG TEXTS USING SEMI-SUPERVISED LEARNING WITH WORD EMBEDDINGS

by
Eda Okur
B.S., Computer Engineering, Boğaziçi University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2015

NAMED ENTITY RECOGNITION FOR TURKISH MICROBLOG TEXTS USING SEMI-SUPERVISED LEARNING WITH WORD EMBEDDINGS

APPROVED BY:

Assist. Prof. Arzucan Özgür (Thesis Supervisor)	. . . .
Assoc. Prof. Banu Diri	. . . .
Prof. Tunga Güngör	. . . .

DATE OF APPROVAL: 15.01.2015

ACKNOWLEDGEMENTS

To begin with, I would like to express my deepest gratitude to my thesis supervisor, Assist. Prof. Arzucan Özgür, for her support and guidance throughout my thesis. I appreciate her very much for being so patient, understanding and kind to me with all her genial attitude. I would also like to thank Prof. Tunga Güngör and Assoc. Prof. Banu Diri for kindly accepting to be in my thesis committee and providing valuable feedback.

I would like to thank fellow graduate student Hakan Demir for his helpfulness and knowledge. I am grateful that he supported me by sharing his valuable ideas that inspired the initiation of my thesis.

I also feel beholden to all researchers who shared their data sets for academic usage and who created publicly available tools in the name of science, which contributed a lot to my thesis. Those researchers are mainly Haşim Sak, Taner Sezer, Anıl Bolat, Gökhan Tür, Gülşen Eryiğit, Dilek Küçük and many more.

I am thankful to The Scientific and Technological Research Council of Turkey (TÜBİTAK), The Science Fellowships and Grant Programmes Department (BİDEB) for providing financial support with 2210 National Scholarship Programme for MSc Students during my Master of Science studies.

I deeply thank my dearest friends for their emotional support and motivation during my extensive working days, and for all pleasant time we shared.

Lastly, my deepest love and gratitude are reserved to my beloved family for their unconditional love and support, and their high confidence and trust in me that always makes me feel stronger.

ABSTRACT

NAMED ENTITY RECOGNITION FOR TURKISH MICROBLOG TEXTS USING SEMI-SUPERVISED LEARNING WITH WORD EMBEDDINGS

Recently, due to the increasing popularity of social media and the value of information contained within real data, the necessity for extracting information from informal text types such as microblog texts has gained significant attention, together with the challenges it brings to the Natural Language Processing (NLP) research community. In this study, we focused on the Named Entity Recognition (NER) problem on informal text types such as microblog texts for Turkish, which is a morphologically rich language. For that purpose, we utilized a semi-supervised learning approach composed of an unsupervised stage followed by a supervised stage based on neural networks. We applied a fast unsupervised method for learning continuous representations of Turkish words in vector space. We make use of these obtained word embeddings, together with language independent features that are engineered to work better on informal text types, for generating a Turkish NER system on microblog texts. For examining informal and short texts in Turkish, we focused on the most popular microblogging environment called Twitter and we evaluated our Turkish NER system on short and unstructured Twitter messages called tweets. With our NER system, we achieved better F-score performances than the published results of previously proposed NER systems on Turkish tweets. To be more precise, we outperformed the state-of-the-art F-score by up to 11% on the same Turkish Twitter data. The only language dependent stage of our system is the normalization scheme we applied for Turkish microblog texts as a preprocessing step before the NER application, which improves the performance of our NER system on informal text types. Since we did not employ any language dependent features, other than text normalization, we believe that our method can be easily adapted to microblog texts in other morphologically rich languages.

ÖZET

TÜRKÇE MİKROBLOG METİNLERİNDE YARI GÜDÜMLÜ ÖĞRENME TEKNİĞİYLE KELİME TEMSİLLERİ KULLANARAK VARLIK İSMİ TANIMA

Günümüzde sosyal medya kullanımının artan popülerliği ve sosyal meydada paylaşılan verilerin içerdiği bilginin değeri göz önüne alındığında, bu tür yapılandırılmamış metinlerden bilgi çıkarımı yapabilmek büyük ilgi görmeye başlamıştır. Bu durum doğal dil işleme araştırmaları açısından pek çok zorluğu da beraberinde getirmiştir. Bu çalışmamızda morfolojik açıdan zengin bir dil olan Türkçe için varlık ismi tanıma probleminin, özellikle mikroblog metinleri gibi yapılandırılmamış metinlerde çözümüne odaklandık. Bu amaçla, güdümlü ve güdümsüz öğrenme aşamalarından oluşan ve yapay sinir ağlarını baz alan yarı güdümlü bir öğrenme tekniği kullandık. İlk olarak hızlı ve güdümsüz bir öğrenme metodu kullanarak çok boyutlu sürekli vektör uzayında Türkçe kelime temsillerini elde ettik. Daha sonra gerek bu kelime temsillerini, gerekse yapılandırılmamış mentinler için daha iyi sonuç verecek şekilde uyarlanmış, dilden bağımsız öznitelikleri kullanarak bu tür metinler için bir Türkçe varlık ismi tanıma sistemi geliştirdik. Yapılandırılmamış ve kısa Türkçe metinleri incelemek amacıyla, en popüler mikroblog platformu olan Twitter üzerine yoğunlaştık ve geliştirdiğimiz sistemi tweet adı verilen kısa Twitter mesajları üzerinde denedik. Sistemimizin Türkçe Twitter mesajları üzerindeki performansının daha önce bu amaçla yayınlanmış sistemlerin performansından daha iyi olduğunu gördük. Türkçe Twitter metinlerinde varlık ismi tanıma için yayınlanmış en gelişkin sistemin F-ölçütü değerini %11 iyileştirme ile aşmış olduk. Sistemimizin dile özgü tek aşaması, varlık isimleri tanınmadan önce Türkçe Twitter metinleri üzerinde uyguladığımız Türkçe metin normalizasyonu aşamasıdır ve bu aşama yapılandırılmamış metinlerde performansı artırmaktadır. Normalizasyon aşaması dışında dile özgü öznitelikleri doğrudan kullanmadığımız için yöntemimizin morfolojik açıdan zengin diğer dillerdeki yapılandırılmamış metinlere de kolayca uyarlanabileceğine inanıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ANNIE	A Nearly-New Information Extraction System
CoNLL	Conference on Computational Natural Language Learning
CRF	Conditional Random Fields
EMM	Europe Media Monitor
GATE	General Architecture for Text Engineering
HMM	Hidden Markov Model
IE	Information Extraction
IR	Information Retrieval
KNN	K-Nearest Neighbors
LDA	Latent Dirichlet Allocation
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
PLO	Person - Location - Organization
POS	Part-of-Speech

1. INTRODUCTION

Micro-blogging environments allow users to post short messages and they provide a new form of communication that increased its popularity in the last decade. Among these, Twitter is the most popular micro-blogging service founded in 2007 with more than 100 million users by 2010, and more than 500 million accounts by 2012. It is a form of social networking site where short messages, or tweets, are shared to the followers of a user in real-time. The most recent statistics by 2014[1] about Twitter usage indicate that there are 271 million monthly active users, 500 million Tweets are sent per day and Twitter supports 35+ languages.

Twitter became an interesting platform for exchanging ideas and thoughts, following recent developments and news, or discussions on any possible topic. Since Twitter has an enormously wide range of users with varying interests and sharing preferences, a significant amount of content is being created rapidly. Therefore, mining such platforms can extract valuable information. As a result, extracting information from Twitter has become a hot topic of research in Information Retrieval (IR) recently.

There are many possible intentions for retrieving information from Twitter. Most of the recent research about Twitter has focused on its network and community structure, like applying link prediction analysis or ranking users on Twitter. A different branch of research has focused on a systematic analysis of the textual content available on Twitter. For the textual analysis side, one popular research area is opinion mining or sentiment analysis, i.e. deciding whether a posted message is positive, negative or neutral. This analysis is surely useful for companies or political parties to gather information about their services and products. Another popular research area for Twitter text mining is content analysis, or more specifically topic modeling. This area is useful for text classification and filtering applications on Twitter. Moreover, event monitoring and trend analysis are also other examples of useful application areas on microblog texts.

[1] https://about.twitter.com/company
[2] http://en.wikipedia.org/wiki/Named-entity_recognition

In order to build successful social media analysis applications, especially on microblog texts, it is necessary to employ successful processing tools for Natural Language Processing (NLP) tasks such as Named Entity Recognition (NER) as a building block for larger analysis applications. NER is an important subtask of Information Extraction (IE) and it is a critical stage for various NLP applications such as machine translation, question answering, and opinion mining. The aim of NER is classifying and locating atomic elements, or words, in a given text into predefined categories like the names of the persons, locations and organizations. For instance, in the area of opinion mining, or sentiment analysis, the sentiment of a given text should be based on named entities such as person names for political popularity analysis, or organization names like companies and products for public relations analysis on social media. Extracted named entities for location names can also be used for event monitoring and disaster detection applications on microblog texts.

NER is accepted as a solved problem in NLP for well-written texts in well-studied languages like English, because state-of-the-art NER systems for English are considered to produce near-human performance[2] on formal text types such as news articles. However, NER still needs further work for morphologically rich languages like Turkish due to its complex structure that brings challenges for the NER task and also due to the fact that language processing tools, data sets, and related sources are relatively rarer for Turkish than for English. Moreover, most of the NER systems in the literature are designed for formal texts such as news articles that have well-formed text structure. The performance of these NER systems drops significantly when applied on real data consisting of the informal text types such as microblog text, or tweets. Due to the increasing popularity of social media and value of information carried within real data, the necessity for extracting information from informal text types like tweets has gained significant attention recently, together with the challenges it brings to the NLP research community.

In the remaining of this chapter, we will present some important background information regarding NER, the Turkish language, and finally Twitter. We will also mention about the challenges of Turkish for NLP and for NER, together with the challenges of Twitter text data, which lack proper structure and formality from the

[2] http://en.wikipedia.org/wiki/Named-entity_recognition

language usage point of view, for NLP and for NER. After that, we will explain our proposed methodology for solving the problem of NER for Turkish on tweets. Lastly, we will list the general outline structure of our thesis study briefly.

1.1. Named Entity Recognition

In this section, we will provide the necessary background information about the NER task that will be useful to understand the rest of our study. We will define the structure of the predefined categories used by most of the NER systems and we will also examine the metrics that are commonly used for performance evaluation of NER systems.

In 1995, the Named Entity Recognition (NER) task is firstly defined in a well-structured manner at the 6th Message Understanding Conference (MUC-6). The aim of the NER task is defined as: given an input text, identify all instances of the elements, i.e. words, which belong to seven defined categories. The NER task is defined as consisting of three subtasks (entity names, temporal expressions, number expressions) so that these seven predefined categories are actually grouped into three subtasks. These seven categories and three subtasks are defined in MUC-6 [1] as follows:

- ENAMEX (Named Entities): This subtask is defined for proper names, acronyms, and various other unique identifiers. Such named entities are categorized via the TYPE attribute as follows:
  - (i) ORGANIZATION: named corporate, governmental, or other organizational entity
  - (ii) PERSON: named person or family
  - (iii) LOCATION: name of politically or geographically defined location (cities, provinces, countries, international regions, bodies of water, mountains, etc.)
- TIMEX (Temporal Expressions): This subtask is defined for absolute temporal expressions only. These tagged tokens are categorized via the TYPE attribute as follows:
  - (i) DATE: complete or partial date expression
  - (ii) TIME: complete or partial expression of time of day

- NUMEX (Number Expressions): This subtask is defined for two types of numeric expressions, namely monetary expressions and percentages, which may be expressed in either numeric or alphabetic form. The complete expression is covered here that is categorized via the TYPE attribute as follows:
  (i) MONEY: monetary expression
  (ii) PERCENT: percentage

In this study, among these seven different predefined categories of MUC-6, our main focus will be on ENAMEX type named entities. These are namely PERSON, ORGANIZATION, and LOCATION. The reason behind this choice is mainly because ENAMEX types are the most informative and thus most popular named entity types, which have commonly been used in most of the NER studies so far. To provide a comparison with the previous studies, we will also examine these three categories of named entities in our study.

MUC-6 [2] and MUC-7 [3] task definitions and evaluation designs have been popular among most of the NER studies, which have used these MUC categories and original guidelines in their NER systems, with minor differences. An example sentence presented in the evaluation design of MUC-6 [2] that is annotated based on the above seven categories can be found in Figure 1.1.

```
Mr. <ENAMEX TYPE="PERSON">Dooner</ENAMEX> met with <ENAMEX TYPE="PERSON">Martin Puris</ENAMEX>, president and chief executive officer of <ENAMEX TYPE="ORGANIZATION">Ammirati & Puris</ENAMEX>, about <ENAMEX TYPE="ORGANIZATION">McCann</ENAMEX>'s acquiring the agency with billings of <NUMEX TYPE="MONEY">$400 million</NUMEX>, but nothing has materialized.
```

Figure 1.1. An example sentence presented in [2] based on MUC-6 annotations.

In addition to the predefined categories of the NER task, the evaluation of NER systems' performances is also an important topic to be clarified here. NER systems are usually evaluated by comparing their output versus the human labeled corpus output, or gold standard. By using different evaluation metrics, we can achieve different evaluation performance results with the same NER system. Therefore, a guideline is also necessary for the evaluation metric to compare different NER systems in terms of performance. Here, we will examine two popular metrics that are commonly used to evaluate the performance of a typical NER system. These are namely the MUC metric and the CoNLL metric, defined as follows:

- MUC metric: MUC defines an F-score on two axes, namely TYPE score that measures the ability to find the correct type and TEXT score that measures the ability to find the exact text. To be more precise, TYPE score considers an assignment as correct if the type of a named entity is assigned correctly by ignoring the boundary of the entity as long as there is an overlap. On the other hand, TEXT score evaluates an assignment to be correct if the boundary of a named entity is detected correctly without taking into account the type of the entity. The overall TYPE and TEXT scores are micro-averaged F-score. The overall MUC score is defined as the average of these TYPE and TEXT scores.
- CoNLL metric: This metric is stricter than MUC metric since it concentrates on finding phrase-level named entities. To be more precise, CoNLL metric evaluates an assignment to be correct only if both the type and the boundary of a named entity are assigned correctly. In this metric, the overall score is calculated by using micro-averaging as well. CoNLL metric is defined in [4] and it is commonly used in most of the recent NER studies.

To understand the differences between these two metrics as scoring mechanism, an example sentence annotated according to MUC guidelines is presented in [5] as follows:

```
    Unlike <ENAMEX TYPE="PERSON">Robert</ENAMEX>, <ENAMEX
TYPE="PERSON">John Briggs Jr</ENAMEX> contacted <ENAMEX
TYPE="ORGANIZATION">Wonderful Stockbrockers Inc</ENAMEX> in <ENAMEX
TYPE="LOCATION">New York</ENAMEX> and instructed them to sell all his
shares in <ENAMEX TYPE="ORGANIZATION">Acme</ENAMEX>.
```

Figure 1.2. An example sentence presented in [5] based on MUC annotations.

Suppose that this sentence with gold labels is given to a NER system as an input and the following output sentence, presented again in [5], is produced:

```
<ENAMEX TYPE="LOCATION">Unlike</ENAMEX> Robert, <ENAMEX TYPE="ORGANIZATION">John Briggs Jr</ENAMEX> contacted Wonderful <ENAMEX TYPE="ORGANIZATION">Stockbrockers</ENAMEX> Inc <TIMEX TYPE="DATE">in New York</TIMEX> and instructed them to sell all his shares in <ENAMEX TYPE="ORGANIZATION">Acme</ENAMEX>.
```

Figure 1.3. Output sentence of a NER system presented in [5] for the input sentence shown in Figure 1.2.

We need to keep three measures in mind in order to evaluate this output sentence. These are the number of correctly annotated answers given as an output by the system (say C), the number of total actual system predictions (say A), and the number of total possible entities annotated with gold labels in the input data (say P). We know that precision shows the number of correct outputs over the number of all outputs produced by the system, whereas recall shows the number of correct outputs over the number of all outputs in the golden data. In our case, precision is computed as C/A and recall is computed as C/P. Note that the final MUC score is defined as the micro-averaged F-score that is the harmonic mean of precision and recall computed over all entity segments, both on TYPE and TEXT axes. We know that F_1-score, or F-score, is defined as twice of the precision times recall over precision plus recall, due to the definition of harmonic mean.

For the example output sentence shown in Figure 1.3 and for the MUC metric, C is 4 (2 TYPE + 2 TEXT), A is 10 (5 TYPE + 5 TEXT), and P is 10 (5 TYPE + 5 TEXT). As a result, precision is 40%, recall is 40%, and therefore the overall F-score for MUC metric is 40%. As you can see, MUC metric gives partial credits for errors occurring on only TYPE or only TEXT axis. Whereas for the CoNLL metric, since only one of the outputs of the system exactly matches with the corresponding entity in the corpus presented by the gold standard, C is only 1. In addition, A is 5 since the system guessed 5 entities, and P is 5 since there exist 5 true entities within the human-labeled data. As a result, precision is 20%, recall is 20% and thus the overall F-score for CoNLL metric is 20%. As you can see, exact-match evaluation metrics like CoNLL metric puts an additional constraint on the systems output to be considered as correct, and therefore CoNLL metric is known to be stricter than MUC metric.

In this study, between these two different metrics, we will be using the CoNLL metric for the evaluation of our NER system's performance since it has become a commonly accepted metric for the NER task recently.

Note that although phrase-level F_1-score, which is a strict one that the CoNLL metric is based on, is a commonly used metric for NER task in general, we will report token-level F_1-score in addition to the phrase-level F_1-score in this study. Token-level F_1-score is the same as what we obtain from the TYPE score component of the MUC metric. It simply gives credit to partial extractions if the entity type is correct, by ignoring the boundaries. Token-level F_1-score is not as strict as phrase-level one for sure, and therefore it is generally higher than the phrase-level one, but the reason we are also reporting this value is that it can be useful for the NER task especially on tweets. We believe that although the boundary of a named entity within a tweet is not 100% correct, determining the correct type for such partial extractions is still important in the microblog domain, since the tweets can be filtered or retrieved easily by using such partial matches. This ability can still be very useful in many social media analysis related applications, and therefore we believe it is profitable to report both phrase-level and token-level F_1-scores for NER systems on Twitter.

1.2. Turkish NLP Challenges

In this section, we will provide the necessary background information about Turkish NLP, especially the NER task for Turkish that will be useful to understand the rest of our study. We will mention the structure of the Turkish language and explain the difficulties and challenges it possesses for the NER task.

Although NER is considered as a solved problem in NLP for well-written texts in well-studied languages like English, it still needs some attention and more work for morphologically rich languages, such as Turkish. This is mainly because the Turkish language has complex structure that possesses extra challenges for the NER task. In addition to this complexity, language processing tools, data sets, and related sources are relatively rarer for Turkish compared to those for English. Therefore, the achieved accuracies of Turkish NER are still behind the reported results for English [6] and there is still room for improvements.

Turkish, as a morphologically rich language, is very different from English, since it is accepted as a highly agglutinative language with an inflectional morphology. Such languages have a common data sparsity problem, also known as lexical sparsity, since in Turkish, it is possible to produce hundreds of different word forms from a single root, especially if it is a verb. This production usually depends on the tense, mood, and the person arguments. This phenomenon is also examined in the work of Hakkani-Tür [7], and it is shown that on the English and Turkish corpora of around 10 million words, the number of unique word forms is 97,734 for English, whereas it is 474,957 for Turkish. On the other hand, if only the root forms are taken into account, the number of unique word forms becomes 94,235 for Turkish. Therefore, it is concluded that from the same root form, on average, five different word forms can be generated in Turkish that indicates the data sparsity problem again. Therefore, in most of the NER studies for Turkish, lemmas or stems are used instead of word forms to resolve this data sparsity problem. This requires the analysis of the morphological structure of the language that is considered for the NER task. Moreover, language specific features should be added to the NER system for the success of most of the NER studies [6].

Another challenge of Turkish for NLP and especially for NER, also stated in [6], is that Turkish is a free word order language. This means that the position of the word within a sentence may not give valuable clue about whether it is a named entity or not. In addition to this difficulty, Turkish person names, especially the first names, are often also used as a common words in a daily language like Deniz (sea), Özgür (free), Barış (peace), Gizem (mystery), Mert (brave), Meltem (breeze), Rüya (dream), Nehir (river) etc. Note that in Turkish, only the proper nouns have the initial letter capitalized, together with the beginning of a sentence. However, this capitalization clue is valid only for formal text types such as newspaper articles. This poses an additional difficulty for Turkish NER, especially on informal texts. Note that we will cover the issues of informal text types for NLP in the next section.

Using gazetteer lists in a dictionary look-up style is a very common phenomenon for a NER task and used by most of the NER systems. A gazetteer usually consists of a set of lists, mostly manually extracted from web, containing common entity names such as person names, location names, organization names etc. and possibly also their indicators. Since the manually annotated data for training is relatively less for morphologically rich languages like Turkish than for English, the usage of gazetteers gains more importance for Turkish because gazetteers may help to overcome the lack of data. However for Turkish, the problem is that these gazetteer lists that are specifically crawled for Turkish names are not publicly available in general for the usage of other NER systems.

1.3. Twitter NLP Challenges

In this section, we will provide the necessary background information about informal text type such as microblog texts, especially about Twitter for NLP and specifically for the NER task. We will mention the structure of Twitter short message texts, called tweets, and explain the difficulties and challenges it poses for the NER task.

There are several different challenges for NLP tasks on micro-blogging environments like Twitter. Kireyev *et al.* [8] summarized these challenges for traditional NLP technologies to be applied on Twitter data. The first challenge is the very short text length. In Twitter, the text size is limited to 140 characters for each tweet. For text classification, short texts contain sparse data and thus it is very difficult to classify them accurately. This limitation brings an additional burden on the NER task, because it is common in tweets to exclude crucial contextual clues such as titles for person names or other useful informative indicators for named entities that can be found in long and formal text types. Since each tweet is treated as a single document with only 140 characters, it is also difficult to make use of non-local features such as content aggregation and prediction history that are used for formal text types mainly composed of news articles, for the NER task on tweets.

An additional challenge for NLP on tweets is the informal structure of the language used on microblogging sites. The language used in Twitter is less formal; therefore the proper grammar rules and punctuations are missing in tweets quite often. Moreover, tweets may contain abbreviations due to text length limitation, together with the Internet slang words such as “omg” (oh my God), “lol” (laughing out loud) etc. In addition, Twitter users may modify words to emphasize their feelings, such as using repeated vowels to increase the strength of their messages. For all these reasons and more, traditional NLP tools often fail to process such informal texts data. Therefore, it is necessary to identify common words and keywords on Twitter that could be useful. In order to defuse such issues of writing and spelling errors, abbreviations, slangs, and special modifications found in tweets, it is recommended to apply a normalization step to correct and expand such usages before using the NER system on tweets.

Especially for the NER task, Twitter introduces interesting challenges due to the local and specific references found in tweets. To be more precise, Twitter messages may refer to specific events, locations and other named entities together with the implied references to locations [8]. In Twitter, using contracted forms and metonymic expressions instead of full organization or location names is very common. For example, users may refer to “Boğaziçi Üniversitesi” (Boğaziçi University) shortly as “boğaziçi” in tweets. It is also common to use single forenames, surnames or even only nicknames instead of full person names in tweets. Therefore, we cannot solely trust the predefined entity lists that are prepared mostly from formal text type data, such as news articles, to work properly on Twitter for NER. Even complex NER methods that work well on formal text types could fail to recognize certain portions of named entities on informal text data like Twitter messages.

Capitalization is another issue for the NER task on tweets. We know that most of the successful NER systems use the capitalization feature as an important clue indicating that the word with capitalized initial letter is a good candidate for a named entity. Although this assumption is mostly valid for formal text types such as news articles where proper capitalization schemes are almost always followed, since the editors revise these formal texts before publication in terms of both grammar and spelling rules, it is not the case for informal text types. Most of the Twitter users do not obey the capitalization rules for person names, organizations and locations in their tweets, since tweets are considered as a form of fast communication that is similar to SMS messages. Therefore, we cannot simply assume that if a word in a tweet lacks proper capitalization, then it is not a good candidate for a named entity by the NER system. This issue is an important disadvantage for informal text types for the NER task, since we cannot benefit from the capitalization clue that plays an important role on the performance accuracy of successful NER systems designed for formal text types.

In addition to lacking proper capitalization for named entities in informal text types, another common punctuation violation seen in Twitter, also mentioned in [9], is not using apostrophes properly to separate person, location, and organization names from their attached suffixes. This makes the NER task even more difficult to apply on informal text types since apostrophe usage is another helpful mechanism for separating proper nouns from suffixes in order to recognize named entities efficiently. This burden requires a NER system to use a successful morphological analyzer to detect named entities that lack apostrophes.

Twitter specific language elements also reveal interesting aspects in terms of NER on tweets. As an example, hashtags and their usage can be important to detect named entities on Twitter. In Twitter, hashtags are special elements in tweets starting with "#" and usually written without whitespaces, which are generally related with specific topics and may include named entities quite often. According to a recent study [9], the appearance of named entities within hashtags is observed in a way that in their data set of tweets, 70 of the total of 153 hashtags have annotated named entities. Moreover, 14 of these 70 hashtags are covered fully by the named entities such as person names. In addition, again in this data set, there are 31 annotated multi-token named entities that are presented like hashtags with no whitespaces. As a result, they concluded that 7.6% of named entities in their tweet data set are found within hashtags or hashtag-like usages with no whitespaces. This trend is also a challenge for a traditional NER system to work properly on tweets, because without considering such Twitter specific cases, the system will miss to recognize named entities that appear in hashtags or hashtag-like usages.

Twitter messages' informality also plays an important role on creating additional challenges for the Turkish NLP tasks. One of the most important challenges of Turkish tweets for NLP, also mentioned in a recent work [10], is a trend of not using specific

Turkish characters that are not found in the Latin alphabet (ç, ğ, ı, İ, ö, ş, ü) and instead, using their equivalent counterparts of universal characters (c, g, i, I, o, s, u) in Turkish tweets. Note that these language specific characters with special marks are known as diacritics. The reason behind this trend of using non-diacritics in Turkish tweets might be the usage of universal English keyboards, mostly on mobile devices, either to type faster or just because it is the default. This usage may cause a typical Turkish NER system to ignore named entities that originally have diacritics but used with non-diacritics versions in tweets. One attempt to alleviate this problem might be instead of using strict lists of person-location-organization names as lexical resources, we can add also non-diacritic versions of named entities into those lists if any. Another approach might be applying diacritic-based normalization on tweets before the NER system. However, this normalization might also be problematic since this trend causes an ambiguity problem due to the fact that these non-diacritic characters that are used instead of their diacritic counterparts are also valid characters in Turkish and a word written in non-diacritic format can have multiple diacritic counterparts with different meanings in Turkish. Moreover, if a named entity with diacritics is used as its non-diacritic variant in a tweet, and if this non-diacritic variant is also a valid word in Turkish, then the diacritic-based normalization will ignore this word and as a result, Turkish NER system on tweets will not recognize this named entity.

In the previous section, we mentioned about the gazetteers and their common usage for the NER task. We stated that using gazetteer lists can be helpful especially when we have comparably less annotated data for training of the NER systems, which can be the case for morphologically rich languages such as Turkish. This problem is especially significant when we focus on Turkish tweets because the amount of manually annotated training data composed of tweets in Turkish for the NER task is much more scarce compared to the English tweets or to the Turkish news articles. Therefore, the usage of gazetteer lists can be a lot more helpful in the case of the NER task on Turkish tweets. However, again the challenge is that these gazetteers manually extracted from the web for Turkish are in general not publicly available for our usage on the NER task for Turkish tweets. Moreover, even if we have public Turkish gazetteer lists prepared mostly for the formal text types, these lists should be further customized for Twitter due to the previously mentioned challenges of informal text types. In addition, these gazetteer lists would require frequent updates, since Twitter is a very dynamic environment in terms of content and the new names of entities will continue to emerge in time to be added into such common dictionary of entity names.

Part-of-speech (POS) tagging is also an important NLP task and it is another sort of labeling task similar to the NER task. POS taggers simply assign to words the correct syntactic category such as verb, noun, adjective and many more in given context. Applying the POS taggers before the NER system and using the outputs of POS taggers as additional morphological features for the NER task is a common phenomena. Although POS taggers usually work with high accuracy for well-studied languages like English, there is also performance degradation for POS taggers for morphologically rich languages like Turkish. Moreover, similar to the NER task, POS taggers that are trained on the formal text types such as news articles and achieved high performance on them usually suffer from performance drop when applied directly on informal text types such as Twitter data. Therefore, achieving a well-performed POS tagger for Turkish that is specialized on Twitter data is another challenging research area and this situation creates an additional difficulty for building a Turkish Twitter NER system that can benefit from a successful Turkish Twitter POS tagger outputs. To the best of our knowledge, there is no such study specialized on Turkish Twitter POS tagging so far that we can make use of its reliable results to further improve the Turkish Twitter NER system.

1.4. Proposal

In order to accomplish the NER task on a challenging domain composed of informal microblog texts in the morphologically rich Turkish language, we adopted a semi-supervised learning approach based on neural networks that employs distributed representations of words. At the first stage, we learnt continuous representations of words in vector space, i.e. word embeddings, by employing a fast unsupervised learning method on a huge unlabeled corpus in Turkish. In the second stage, we exploited Turkish word embeddings together with language independent features and trained our neural network on annotated data. Finally, we evaluated our NER system on annotated Turkish Twitter messages with named entities. We have compared our results on two different Turkish Twitter data sets with the state-of-the-art NER system proposed for Twitter data in Turkish and we have shown that our Turkish Twitter NER system outperforms the state-of-the-art performance results by up to 11% and 9% in terms of F-score performance on these two Turkish Twitter data sets.

Our proposed NER system is adapted to perform better on microblogging texts in Turkish with various design choices. At the unsupervised stage, in addition to using a huge corpus composed of Turkish news articles, we also explored using large amount of data composed of Turkish tweets in order to learn Turkish word embeddings. Note that these word embeddings are used as an important feature at the supervised stage and hence, we propagated the specific language usages in tweets indirectly to our supervised stage and trained our NER system with that additional knowledge. Moreover, we applied Twitter processing on our tweets data in order to tag Twitter specific usages such as mentions, hashtags, smileys, URLs etc. Exploiting Twitter specific keywords into our large Turkish tweets corpus makes our system specialized towards informal microblog texts. Moreover, we investigated the usage of the capitalization feature. We have shown that in order to have a NER system that performs better on informal text types where we usually lack proper capitalizations; the capitalization feature should be turned off. Furthermore, we applied a Turkish text normalization scheme specially designed for social media data on our tweets and we have shown that it also improves the performance. This text normalization scheme involves tagging certain Twitter specific keywords such as hashtags, mentions, retweets, smileys, URLs etc. which makes our final NER system specialized towards microblog texts together with the similar processing performed on the unlabeled Twitter corpus at the unsupervised stage. Since this normalization attempts to recover the lacking capitalizations in tweets, we have shown that if you have a normalization stage before applying NER on your data, the capitalization feature should be turned on again. The last adaptation for microblogging environments we examined is training our NER system on annotated Turkish tweets. We have shown that training the NER system even with limited amount of annotated tweets leads to promising results.

We have shown that utilizing the word representations in a semi-supervised learning approach is highly effective for Twitter NER and can result in state-of-the-art performance even without using any language dependent features and gazetteer lists. Since other than the normalization scheme we applied, we have not used any language dependent features to build our Turkish Twitter NER system, we believe that our approach can be easily adapted to other morphologically rich languages.

Finally, the main contributions of our study can be summarized as follows:

- We outperformed the previous state-of-the-art performance for Turkish Twitter NER by up to 11% improvements in terms of phrase-level F-score.
- A neural network based semi-supervised learning approach is adapted for the NER task on Turkish microblog texts for the first time in the literature.
- We explored the usage of distributed word representations for the first time in Twitter NER for Turkish.
- Using an additional large Turkish tweets corpus is proposed to learn more Twitter specific Turkish word embeddings for Turkish NER on tweets, which is shown to be useful for the first time.
- We compared the effects of using word embeddings versus text normalization and we have shown that usage of word embeddings yields better NER performance results than the recently proposed social media text normalization for Turkish NER on Twitter.
- Unlike the previous Turkish NER models on tweets, we experimented with training on Twitter data instead of news articles.
- Except the normalization stage, which is optional and is not the main reason for our state-of-the-art performance, our Twitter NER system is language independent and hence can be applied to other morphologically rich languages for the NER task on microblog texts.
- We will make our attained Turkish word embeddings, both from Turkish web corpus and Turkish tweets corpus, publicly available for future usages in various NLP tasks for Turkish. We will also make our Twitter NER system for Turkish as a public tool for further research such as sentiment analysis on Turkish tweets.

1.5. Outline

In this first chapter, we presented an introduction to our study by explaining the motivation and the problem, together with providing necessary background information and then describing our proposal for this problem of NER for Turkish on Twitter. Here we will give a brief overview of the remainder of this thesis. In the next chapter, we will examine the related work presented in the literature so far. In Chapter 3, the model and the architecture of our system will be described in detail. The information about the sources and the analytical settings of the data sets we used will be given in Chapter 4. In Chapter 5, the experiments we conducted and their results will be presented in detail. Finally, the thesis will be concluded with further discussions and future work in the last chapter.

2. RELATED WORK

In this chapter, we will summarize the previous works that are closely related to our study. In the first section, we will mention the Named Entity Recognition (NER) studies within the area of Natural Language Processing (NLP) by examining the previous studies on Turkish NER. Note that until now, these NER studies are mainly for formal texts domains such as news articles for Turkish. In the second section we will summarize recent and important studies on NER for informal texts domains such as microblog texts, especially for Twitter, and mostly for English. After that, we will narrow down our scope of interest in the third section and we will examine the existing studies on Twitter NER for Turkish. Finally, we will explain the definition of word representations and their usage in NLP tasks in the last section.

2.1. Turkish Named Entity Recognition

Turkish is a morphologically rich and also highly agglutinative language. Such morphologically rich languages usually bring out interesting challenges for NLP tasks such as NER. In this section, we will explain the significant previous studies on NER for Turkish.

Tür *et al.* [11] presented the first comprehensive study on NER for Turkish. In this work, they studied different information extraction tasks; one of them was name tagging that we are interested in. Note that within NER, finding only the names is called name tagging. They used four different feature sources as models for this task of tagging names. These models were the lexical, contextual, morphological and name tag models. The lexical model captures lexical information using only word tokens. Then, the contextual information using the surrounding context of the word tokens is captured by the contextual model. Then, the morphological information with respect to the corresponding case and name tag information is captured by the morphological model using the morphological parses of the words. Finally, the name tag model captures the name tag information. They used HMM (Hidden Markov Model) based model for all these four stages. For the evaluation, they used MUC metrics and general news text as a domain. With ENAMEX type only, their F-score performance is stated as 91.56%.

Küçük and Yazıcı [12] proposed a hybrid named entity recognizer for Turkish by incorporating statistical methods to their rule based NER system for Turkish, which was presented previously in Küçük and Yazıcı [13]. They used ENAMEX, TIMEX, and NUMEX entity types. However, they did not provide the performance scores for each of these three entity types, therefore we do not know their performance on ENAMEX entity types only. They have raised the F-score performance on general news text domain to 90.13% from 87.96%, which was stated in their previous work [13]. However, their results are not comparable with similar works, since they used an evaluation metric that gives more credit to partial matches and it is different from the CoNLL and MUC metrics. This study is important since it compares the different domains for Turkish NER for the first time. They used general news text, financial news text, historical text, and child stories to evaluate their system.

Tatar and Çiçekli [14] presented an automatic rule learning system for Turkish NER that exploits different features of text. In this study, the features used are grouped as lexical, morphological, contextual and orthographic features. For lexical features, in addition to the tokens themselves, they also used gazetteer information (e.g. list of person names, list of city names) of words mapped to the tokens. Here they used two-level gazetteer hierarchy where the first level corresponds to each named entity class (e.g. Person, Location etc.) and the second level details the gazetteer categorization (e.g. Location.Country, Location.City etc.). Morphological parses of the words are used as morphological features and the information captured in the surrounding text of the named entities is used as contextual features. For orthographic features, they selected four primitive features as Capitalization (Lower, Upper, Proper, Unclassified), Length Class (Short, Middle, Long), Length (the length of the token), and Type Class (Alpha, Numeric, Alphanumeric, Punctuation, Unclassified). Their evaluation strategy of looking for exact matches is compatible with the CoNLL metric, even though they did not state that they used the CoNNL metric. Their domain of text is based on terrorism news for evaluation. They achieved 91.08% F-score on ENAMEX and TIMEX types, and this score is calculated as 90.63% for the ENAMEX types only.

Yeniterzi [15] introduced using Conditional Random Fields (CRF) for the first time for Turkish NER and utilized the effect of morphology for this task. She constructed two models, namely word-level and morpheme-level models, both based on CRF. The word-level model is the commonly used sequence of words representation that takes into account the word token itself, the root forms of the words, the Part-of-Speech tags of the words, whether the word is proper noun or not, and whether the word has uppercase initial letter or not. In the morpheme-level model, to see the effect of morphological tags, the following features are used: the actual root of the word and morphemes of the word, the Part-of-Speech tag of the root and the morphological tag for the morphemes, the root-morph feature which differentiates roots and morphemes, the proper-noun feature and the case feature of the word. The same training and test data based on the general news domain is used as in [11]. The results are given in CoNLL metric for ENAMEX types and an F-score performance of 88.71% is reported with the word-level model, whereas an F-score of 88.94% is obtained with the morpheme-level model.

Şeker and Eryiğit [6] presented the current state-of-the-art work for Turkish NER. This CRF-based approach proposed for general news text is reported to outperform the previous proposals in the literature on the ENAMEX types for Turkish NER. Their model makes use of morphological, lexical, and gazetteer look-up features. The morphological features are listed as stem of the word, Part-of-Speech tag of the word, noun case of the word that states whether it is non nominal or one of the eight values for nominal, whether the word is proper noun or not, and all inflectional tags after the POS category. The lexical features are the case feature that carries the information about lowercase and uppercase letters used in the token, and the start of the sentence feature indicating the token is the beginning of a sentence or not. Gazetteer look-up features are binary features indicating whether the current token is present in the corresponding gazetteer lists or not. Note that they prepared two kinds of gazetteers called base and generator gazetteers. Their base gazetteers are large for first names, surnames, and location names, whereas their generator gazetteers are relatively small for location, organization, and person names. Again, the same training and test data based on the general news domain is used as in [11] and [15]. They used both CoNLL and MUC metrics for evaluation on the ENAMEX types. In CoNLL metric, they achieved an F-score of 89.59% without gazetteers and 91.94% with gazetteers. In MUC metric, an F-score performance of 92.83% without gazetteers and 94.59% with gazetteers is reported.

Demir [16] conducted a very recent study on NER for morphologically rich languages, namely Turkish and Czech, which is then published by Demir and Özgür [17]. For this task, they used a semi-supervised learning approach based on neural networks. In the unsupervised stage, they adopted a fast unsupervised method for learning continuous vector representations of words and these representations are fed to the supervised stage as additional features. This study is quite different from the previous proposals, since their system does not exploit any language dependent features and as a result, this system can be adapted to other morphologically rich languages easily. It is also shown that word representations obtained very fast by following the approach of Mikolov *et al.* [18] are very useful for NER. Again, the same training and test data based on the general news domain is used as in [6], [11] and [15]. In this work, the CoNLL metric is used for the evaluation on ENAMEX types. They achieved an F-score performance of 91.85% for Turkish without using gazetteers and any language-specific features. It is important to note here that even without using language dependent features such as morphological features, this system outperforms the previous state-of-the-art system [6] that achieved an F-score of 89.59% without gazetteers in CoNLL metric for ENAMEX types on the same training and test data.

2.2. Named Entity Recognition for English Microblog Texts

Named Entity Recognition for English is a well-known and deeply studied area in NLP, and there are many significant studies in the literature for NER. However, the NER studies on microblog texts are relatively limited and quite recent. Surely, Twitter is the most popular and widely used microblog site and extracting information from tweets becomes more and more valuable these days. In this section, we will summarize important recent studies for the NER task on Twitter data for English.

Ritter *et al.* [19] presented a two-phase NER system for tweets, namely T-NER. They used Conditional Random Fields (CRF) for named entity segmentation. In addition, they employed labeled topic modeling by using labeled Latent Dirichlet Allocation

(Labeled-LDA) [20] to utilize the open-domain database of Freebase dictionaries as a source of supervision for subsequent named entity classification. Note that in this study, not only NER but actually a pipeline approach is used that performs first tokenization and then POS tagging and then chunking before using topic models to extract named entities. They used Brendan O'Connor's Twitter tokenizer [21] for English, and implemented their own POS tagger for Twitter called T-POS, and their own chunker called T-CHUNK that performs shallow parsing on tweets. They also built a capitalization classifier, T-CAP, to determine if capitalization is informative or not within a tweet. For NER, they used orthographic, contextual, and dictionary features where their dictionary is composed of a set of type-lists gathered from Freebase. They also used Brown clusters [22] and outputs of T-POS, T-CHUNK, and T-CAP while generating features. For these tweet-specific NLP tools, they included tweet-specific features such as retweets, usernames, hashtags and URLs. It is reported that T-NER achieved an F-score performance of 59% on PLO (Person, Location, Organization) entity types.

Liu *et al.* [23] proposed a hybrid NER approach on tweets that is based on K-Nearest Neighbors (KNN) classifiers and linear CRF labeler, which are applied sequentially under a semi-supervised learning framework. Their KNN classifier gathers global evidence among recently labeled tweets to manage word level classification and their CRF labeler utilizes information from a single tweet and also from the gazetteers. As for the features, their KNN classifier extracts bag-of-words features and their CRF model extracts orthographic, lexical, and gazetteer related features. For evaluation, they used PLO types plus the PRODUCT entity type and reported to achieve an F-score performance of 80.2%.

In their later study, Liu *et al.* [24] this time presented a factor graph-based method that jointly performs named entity normalization (NEN) and NER for Twitter, which allows these two different tasks to reinforce each other. For orthographic features they used whether a word is capitalized or not, alphanumeric or contains any slashes, stop word or not. They also used the word prefixes and suffixes as features. For lexical features they used the lemmas and POS tags of the previous, current and next words; whether the current word is an out-of-vocabulary word; whether it is a hashtag, a link, or a user account. For gazetteer related features, they manually constructed a named entity dictionary from different sources such as Wikipedia, Freebase, news articles and gazetteers that was used in their previous work [23]. For evaluation, again they used PLO types plus the PRODUCT entity type and reported to achieve an F-score performance of 83.6% this time.

Li *et al.* [25] described an unsupervised approach that performs Named Entity Recognition by means of only discovering the presence of named entities regardless of their types for targeted tweets, namely TwiNER. It is a two-step approach, which does not depend on any linguistic features. The first step is partitioning tweets into valid segments of phrases using global context gathered from the World Wide Web, by using resources like Wikipedia. The second step is applying a random walk model on this segment graph, which exploits the local context in tweets. Although this system is only targeted on recognizing all possible named entities in a given Twitter stream and does not care about categorizing the types of named entities, it is still a notable study because even though it employs an unsupervised method, TwiNER is reported to achieve a comparable F-score performance with the supervised systems when applied on the same data.

Oliveira *et al.* [26] proposed a filter-based approach for NER on Twitter data, namely FS-NER (Filter Stream NER). Their system is characterized by the use of lightweight filters that process tweets. Five different filters cover the features they used as follows: term, context, affix, dictionary, and noun filters. The term filter estimates the probability of the given term being an entity. The context filter captures unknown entities by examining the surrounding terms. The affix filter employs the fragments of an observation to decide whether it is an entity or not. The dictionary filter exploits a list of names of correlated entities to understand if the given term is an entity. The noun filter only takes into account terms with capitalized first letter to infer whether it is an entity. Note that all these filters are independent from the grammar rules, which makes their system language-independent. In this study, they defined seven entity types of their own for evaluation purposes and it is reported that FS-NER performs on average 3% better than their baseline based on CRF. Moreover, FS-NER's ability to process large amounts of data in real-time is claimed to make it more practical.

Lastly, Bontcheva *et al.* [27] described an NLP pipeline for an information extraction system specialized for microblog texts, namely TwitIE. It is actually an adaptation of the general purpose ANNIE component of the GATE NLP framework. They took the name gazetteer lookups and sentence splitter modules directly from ANNIE without any modification for Twitter, but adapted all the other components for Twitter including the language identifier, tokenizer, normalizer, POS tagger, and finally the named entity recognizer. Their tweet tokenizer considers URLs and abbreviations as one token, hashtags and user mentions as two tokens with a separate annotation, including the preserved capitalization and orthography features. Their tweet normalizer employs a combination of generic and social media specific spelling correction dictionaries for identifying and correcting errors by using distance heuristics. Their Twitter POS tagger is an adaptation of Stanford POS tagger [28] by training on tweets and by including additional tag labels for URLs, retweets, hashtags, and user mentions. Finally, the NER component of TwitIE benefits from the Twitter-specific modifications in the earlier components as we mentioned, especially their POS tagger. This system is stated to outperform T-NER [19] by achieving an F-score performance of 80% on PLO entity types.

2.3. Named Entity Recognition for Turkish Microblog Texts

We have examined the important studies on Turkish NER and also significant recent studies on Twitter NER, mostly for English. In this section, we will mention about previous studies on the intersection of these two, namely on Turkish Twitter NER. To the best of our knowledge, there exist only three studies addressing NER on Twitter for Turkish, which are published only within the last two years.

Çelikkaya *et al.* [29] presented the first study for Turkish NER on social media data, such as microblog texts like Twitter. They adopted a CRF-based NER system that was presented by Şeker and Eryiğit [6] and evaluated this system on informal Turkish data from different domains such as forum data, speech data and Twitter data for the first time. Note that this system is trained on formal data of news articles as it was before, but tested on informal data of tweets this time. An important part of this initial study is that they prepared a manually annotated data set of Turkish tweets for testing, which contains about 5K tweets and 54K tokens and it includes ENAMEX, TIMEX, and NUMEX entities. The number of annotated PLOs is reported as 1,336. They used the same features such as morphological, lexical, and gazetteer lookup features stated in [6]. Additionally, for the informal texts domain, they created their own text normalizer and processed the input with this before the other stages. This normalizer works on slang words, repeated characters, hashtags, mentions, smiley icons, vocatives, emo style writings, and capitalization. This initial study is reported to reach an F-score performance of 19.28% in CoNLL metric with ENAMEX entity types on their Turkish tweets data.

Küçük *et al.* [9] proposed the second study in the literature on NER for Turkish with Twitter data. This study is in fact an adaptation of a multilingual rule-based NER system of the Europe Media Monitor (EMM) that processes news articles, presented before by Pouliquen and Steinberger [30], for Turkish. Here, a new data set of 2,320 tweets in Turkish is annotated manually with named entities and presented as a public data set. Their annotation includes ENAMEX, TIMEX, and NUMEX entity types, plus a MISC type that stands for names of TV programs / series, movies, music bands and products. Their evaluation is however only on the ENAMEX types, or PLOs for comparison and this publicly available data set contains around 2.3K tweets, 21K tokens and 980 annotated PLOs. They did not give any credit to partial extractions, so their results are comparable to the CoNLL metric. This initial adaptation is stated to achieve an F-score performance of 37.24% on their new tweet data set, and 29.64% on the tweet data set which was presented before by Çelikkaya *et al.* [29]. As an improvement, they extended the resources of this rule-based NER system with frequently used organization and person names found in Turkish news articles. This attempt raised their final F-score to 42.68% on their new Twitter data set, and to 36.11% on the Twitter data set of Çelikkaya *et al.* [29].

In their later study, Küçük and Steinberger [10] proposed some improvements for NER on Turkish Twitter data. In this study, again they used two different Turkish Twitter data sets, one of them is presented in their previous study [9], which they call Twitter Set-1, and the other one is based on the data set presented by Çelikkaya *et al.* [29], which they call Twitter Set-2. Here they used a rule-based NER system that was presented before for Turkish news articles by Küçük and Yazıcı [13]. This system exploits lexical resources like PLO lists and patterns for the named entity extractions, and it includes a simple morphological analyzer together with a configuration of using capitalization clue or not. Here they evaluated their results with both strict metrics where partially extracted named entities are given no credit, which is comparable to the CoNLL metric, and partial metrics where partial extractions take credit as well. They also evaluated the overall results only for PLOs, or ENAMEX types, as well as for 7 types including the ENAMEX, TIMEX and NUMEX entity types. Here we will discuss their results for PLOs only obtained using the strict CoNLL like metric. The initial system without any improvements attempts reached an F-score performance of 46.55% on Tweet Set-1 and 30.19% on Tweet Set-2. Their first improvement is relaxing the capitalization constraint of the rule-based NER system they used. By setting the capitalization feature off, the system regards all tokens as possible Named Entity (NE) candidates without checking whether their initial character is capitalized or not. With this improvement, they achieved an F-score of 47.76% on Tweet Set-1 and 36.63% on Tweet Set-2. The second improvement is done by diacritics-based expansion to their lexical resources used by this rule-based NER system. They simply expanded the entries to include both diacritic and non-diacritic variants in their lexical resources. With this improvement, together with the capitalization feature off, they obtained an F-score performance of 48.13% on Tweet Set-1 and 38.01% on Tweet Set-2. Their last attempt for improvement is exploiting simple tweet normalization by decreasing consecutively repeated characters in words within tweets, by using the list of Turkish unique words of Zemberek [31] for double-check. This last attempt, together with the relaxation of the capitalization constraint, is evaluated only on Tweet Set-1 and achieves an F-score of 47.92% with the strict metrics. They did not combine all these three improvements attempts together for evaluation, so we cannot conclude that this simplistic normalization is really an overall improvement or not.

2.4. Distributed Word Representations

Vector space distributed representations of words are helpful for learning algorithms to reach better results in many NLP tasks, since it provides a method for grouping similar words together. The earliest studies on distributed representations date back to 1986 by Hinton *et al.* [32] and by Rumelhart *et al.* [33]. Distributed representations are defined by Hinton *et al.* [32] such that given a network, entities can be represented by a pattern of activity in a large number of units where each such unit is representing a micro-feature of the entity. The benefit of this type of representation is that, it provides efficiency for using the processing abilities of networks of simple, neuron-like units. Within the same year, Rumelhart *et al.* [33] described a new learning method called back propagation for the first time in networks of neuron-like units. This study was the earliest usage of distributed representations and it provides an efficient way of learning distributed representations using back propagating errors.

The idea of using distributed word representations in vector space is applied to statistical language modeling for the first time by using a neural network based approach with a significant success by Bengio *et al.* [34]. The approach is based on learning a distributed representation of each word, which is also called a word embedding. Each dimension of a word embedding represents a hidden feature of this word and is used in order to capture this word's semantic and grammatical properties. In this study, they used these learned distributed feature vectors for words, which are basically continuous real vectors, in order to capture the similarities between words. One of the most important results obtained from this study is that, significantly better results for statistical language modeling can be achieved when using the neural network as an underlying predictive model for learning distributed word representations, compared to the best of the n-gram based models.

A similar approach is presented by Collobert and Weston [35] later on by using neural networks to share distributed word representations across various NLP tasks such as Part-of-Speech Tagging, chunking, Named Entity Recognition, semantic role labeling, and syntactic parsing. Here, a neural language model is proposed which is faster to train than traditional neural language models, which were slow due to the complexity that is proportional to the size of the vocabulary for every computation. This faster approach, similar to the one in [34], allows making use of a large unlabeled corpus for learning distributed word representations. Later on, Collobert *et al.* [36] again proposed to use distributed word representations together with the supervised neural network and by suggesting this semi-supervised method, they achieved state-of-the art results for the above NLP tasks. Note that they reached state-of-the-art results for English NER as well using this semi-supervised approach.

Although these approaches presented in [34-36] are faster than the traditional approaches of learning distributed word representations, and although it allows a scaling to large corpora, it is still suffering from very long training times of the deep neural networks with many layers. To illustrate the speed challenge, note that it can take weeks to get distributed word representations using these methods. Later on, Mikolov *et al.* [18] showed that continuous vector representations of words could be obtained much faster without sacrificing from the performance. In [18], a similar approach is used with neural networks as presented in [34]. In order to fasten the process of generating word embeddings, the non-linear hidden layer is removed from the architecture since it was the main source of the slowness during the training. This efficient method presented in [18] uses the current word as an input to the projection layer in the neural network and it attempts to predict the words before and after the current word within a certain range. The increased speed is reported being in the order of billions of words per hour. Therefore, we can extract word embeddings within a few hours instead of a few weeks now, which is a significant improvement. It is shown that although this efficient method presented in [18] is much more simplistic than the ones presented in [34] and [36], the obtained word representations are better than [34] and [36] in terms of performance for various NLP tasks. By increasing the efficiency of extracting continuous word representations, Mikolov *et al.* [18] allowed us to use high quality word vectors that are promising to become an important building block for many NLP applications.

3. NAMED ENTITY RECOGNITION FOR TURKISH MICROBLOG TEXTS

In this chapter, we will present the details of our methodology of using semi-supervised learning with word embeddings for the task of Named Entity Recognition for Turkish microblog texts, i.e. on Turkish Twitter data.

Semi-supervised learning is a type of learning technique that makes use of unlabeled data in addition to the labeled data at the training stage. Typically, a large amount of unlabeled data is used together with a relatively small amount of labeled data in the semi-supervised learning approach. It is known that using unlabeled data in addition to the smaller amount of labeled data may provide significant improvement in terms of learning accuracy.

In this study, we present a neural network based architecture consisting of two stages. The first stage is the unsupervised stage, which benefits from the large amount of unlabeled data. The second stage is the supervised stage where a comparably smaller amount of labeled data is used for training the model and testing its performance. The description of our system in details can be found in the following sections.

3.1. Unsupervised Stage

In the unsupervised stage, our aim is to learn distributed word representations, or word embeddings, in continuous vector space. If we can obtain distributed representations of words with a high quality, then we can expect that in this continuous vector space, semantically similar words will be close to each other. This will provide a useful additional knowledge base in terms of our NER task for Turkish microblog texts.

In order to achieve good representation of words, we need a huge amount of unlabeled Turkish corpus to feed the system in the unsupervised stage. The reason we are employing an unsupervised stage as a first stage of our system stems from the known fact that, as word vectors can be trained on large unlabeled text corpus, they can provide generalization for NLP systems trained with limited amount of labeled data in the supervised stage.

A word representation is usually a vector associated with each word, where each dimension of this vector is actually representing a feature. The value of each dimension is defined to be representing the amount of activity for that specific feature. To represent a word as a vector, we can use either a local representation or a distributed representation.

The traditional approach for representing words as vectors is known as one-hot vector representation where only one of the vector elements is active for each word. If we have a vocabulary of size |V|, then each word is represented as a vector of size |V| where only the index of this word is one and the rest is zero. This representation is known as local representation. Although one-hot vector representation of words is easy to interpret, it is not possible to show correlation between two different words with this local representation, and hence it is not very useful for the NER task. In addition, parameters estimated with one-hot representation will be poor for rare words in the corpus. Furthermore, the local representation model lacks the treatment for out-of-vocabulary words that are not found in the corpus.

In contrast, the distributed representation is representing each word as a dense vector of continuous values, or real values, instead of discrete values of one and zero. This vector is a lower dimensional vector where each dimension represents a latent feature for a word. By having lower dimensional dense vectors, and by having real values at each dimension instead of having only a single one and |V|-1 zeroes, distributed word representations are helpful to solve the sparsity problem. On the other hand, distributed word representations are trained with a huge unlabeled corpus using unsupervised learning. If this unlabeled corpus is huge enough, then we expect that distributed word representations will capture the syntactic and semantic properties of each word and this will provide a mechanism to obtain similar representations for semantically and syntactically close words. For all these reasons, we will use the distributed word representations in continuous vector space, which is also called word embeddings, instead of the local representations of words in our study.

In this study, we make use of the publicly available tool presented by Mikolov *et al.* [18], which is called word2vec[3], to obtain the word embeddings. The neural network approach proposed in [18] is similar to the feed-forward neural networks presented before by Bengio *et al.* [34] and Collobert at al [36], where we have four layers including the input layer, projection layer, hidden layer, and output layer. To be more precise, the previous words to the current word are encoded in the input layer and then projected to the projection layer with a shared projection matrix. This projection is actually the concatenation of the feature vectors of the previous words. After that, the projection is given to the non-linear hidden layer and then the output is given to the softmax function in order to a receive probability distribution over all the words in the vocabulary. In such a model presented in [34] and [36], the computation between the projection layer and the hidden layer is complex due to the fact that the values are dense and real. On the other hand, the method used in [18] is much faster in terms of training times since the non-linear hidden layer is removed from the architecture, which was the main cause for the complexity and hence very long training times. This long training time was creating a limitation for the amount of unlabeled data to be used as an input to extract good word embeddings. It is shown that more training data of unlabeled texts improves the quality of the obtained word vectors, because as the unlabeled data size increases, we can obtain more representative feature vectors of words. Therefore, removing the non-linear hidden layer and making the projection layer shared by all words is a significant improvement in [18], which allows us to use a larger unlabeled corpus in the unsupervised stage and thus obtain better word embeddings.

3.1.1. Training Distributed Representations of Words

In this section, we will describe the model architectures for efficiently learning distributed word representations in continuous vector space from a huge amount of unlabeled data. Mikolov *et al.* [18] proposed two model architectures to learn word embeddings with minimized complexity. These two new log linear models both remove the non-linear hidden layer and compensate the representative power loss of word embeddings by increasing the possibility to be trained on much larger data.

[3] https://code.google.com/p/word2vec/

The first model for generating word embeddings in an efficient manner, proposed in [18], is called continuous bag-of-words model, where the hidden layer is removed and the projection layer is shared for all words, not just the projection matrix. As a result, all words from a certain range, i.e. window, are projected into the same position in this model. The weight matrix between the input and projections layers is shared by all inputs. It is called bag-of-words model since the projection is independent of the order of words and it is named as continuous bag-of-wards since it makes use of continuous vector space distributed representations of the context. Here, a log linear classifier is used to classify the current word given the past and future of the word in certain window size. Hierarchical softmax is also used to increase the efficiency further.

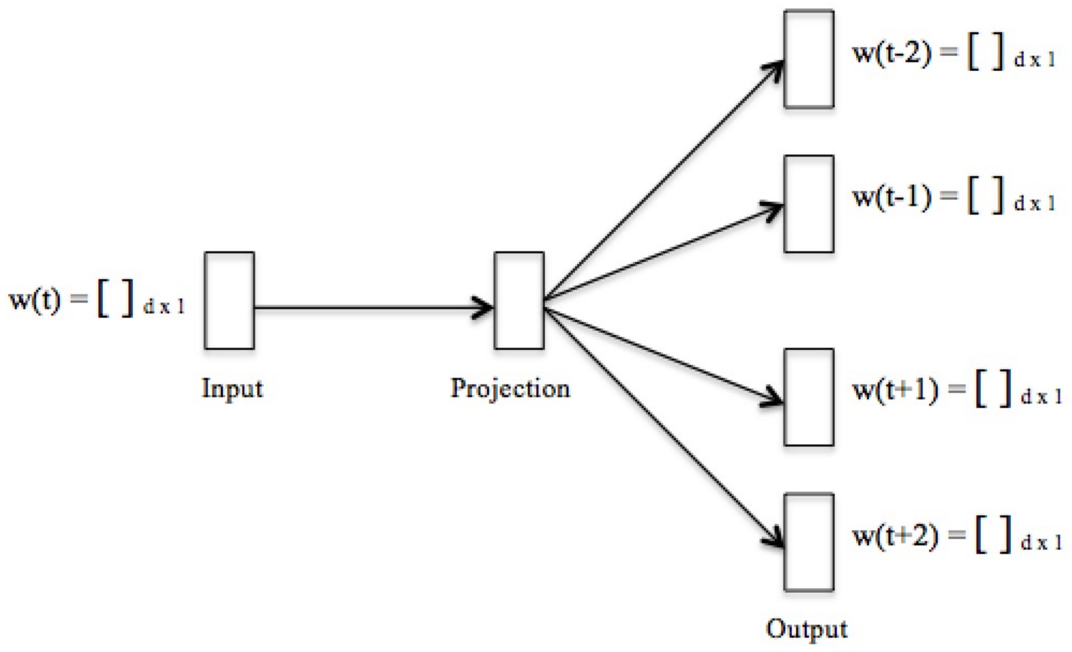

Figure 3.1. Skip-gram model architecture where the aim is to learn the continuous vector representations of words in order to predict the surrounding words [18].

The second model proposed in [18] for learning word representations efficiently is called continuous skip-gram model that is very similar to the first model. The difference is that instead of predicting the current word based on context as in the first model, the skip-gram model aims to maximize the classification of a word based on the other words within the sentence. The skip-gram model uses the current word as an input to the projection layer with a log-linear classifier and it attempts to predict the representation of the neighboring words within a certain range before and after the current word. Mikolov *et al.* [18] showed that the continuous skip-gram model is better than the continuous bag-of-words model in terms of obtaining semantic representations of words. In our study, we also used the continuous skip-gram model in order to obtain better representations of words for Turkish. The skip-gram model architecture we used in our study is shown in Figure 3.1 where the range is taken as 5 so that the previous and next 2 words are predicted from the current word.

The goal of training the skip-gram model is to obtain continuous word representations that can be used to predict the neighboring words within a sentence. To be more precise, given a sequence of input words w_1, w_2, …, w_T, the objective function of the skip-gram model is to maximize the average log probability [37]

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{j=-c\\ j\neq 0}}^{c}\log p\left(w_{t+j}\middle|w_t\right) \tag{3.1}$$

where c is the size of the training range of context and T is the number of words in a given sequence. Here we are calculating the sum of the log probabilities of the previous and next c words of a given word, and then summing them up for all words in the given sequence.

The skip-gram model defines the above inner probability, i.e. the probability of word w_i given w_j, using the softmax function as [37]

$$p\left(w_i\middle|w_j\right) = \frac{\exp\left(u_{w_i}^{T}\, v_{w_j}\right)}{\sum_{k=1}^{V}\exp\left(u_k^{T}\, v_{w_j}\right)} \tag{3.2}$$

where u_w and v_w are the input and output vector representations of word w, respectively, and V is the number of words in the vocabulary. The above formula in not practical to compute since the cost of computing the denominator is proportional to V which is the number of all words in the vocabulary and hence very large.

In order to compute the probability $p(w_i|w_j)$ efficiently, using hierarchical softmax instead of full softmax is proposed in [38]. In the hierarchical softmax, the output layer is represented as a binary tree having V words at its leaves and therefore the complexity is reduced logarithmically since instead of evaluating V output nodes in the neural network, we need to evaluate only $\log_2(V)$ nodes with the hierarchical softmax approach. In this model, each word w can be reached by a certain path from the root of this binary tree. The probability of a word w given w_I is defined as [37]

$$p(w|w_I) = \prod_{j=1}^{L(w)\,-\,1} \sigma\left([\![n(w,j+1) = ch(n(w,j))]\!] \cdot u_{n(w,j)}^{T} v_{w_I}\right) \tag{3.3}$$

where $\sigma(x) = 1/(1 + \exp(-x))$, $n(w,j)$ is the j^{th} node on the path from the root to w, and $L(w)$ is the length of this path. For any inner node n of the tree, $ch(n)$ is defined to be an arbitrary fixed child of n and $[\![x]\!]$ is 1 if x is true and -1 otherwise. The fact that $\sum_{w=1}^{V} p(w|w_I) = 1$ can be verified [37]. With this hierarchical softmax formula, we have one representation v_w for each word w and one representation u_n for every inner node n of the tree. In this model, backpropagation error is used together with stochastic gradient descent to learn high-quality word representations efficiently.

While training the distributed word representations in continuous vector space using a large unlabeled corpus, the chosen dimension of the vectors is important as well as the predefined range of the surrounding words. As the desired vector dimensions increase, which is denoted as d in Figure 3.1, the representativeness and quality of the obtained word embeddings improve together with a cost of complexity. Similarly, as the predefined range of neighboring words increases in order to predict the current word, the prediction accuracy improves again with an additional cost of computational complexity [37]. In this study, for our NER task on Turkish Twitter data, we have chosen 200 as the dimension of the obtained word vectors. The range of the surrounding words is chosen to be 5, so that we will predict the distributed representations of the previous 2 words and next 2 words using the current word. This vector size and range decisions are aligned with the choices made in a previous study [17] for the NER task on Turkish structured data.

Another important aspect of training word embeddings in continuous vector space with skip-gram model is the amount of unlabeled corpus used for training. As discussed in [37], using large amount of data for training is crucial in terms of the quality of the word representations obtained and it can easily affect the resulting accuracy of various NLP tasks that make use of these word embeddings. Mikolov *et al.* [18] used a data set of size 1.6B words for English to train word embeddings. For the Turkish NER task on news domain, Demir and Özgür [17] used a Turkish corpus of 1.02B words in order to obtain word embeddings. Since this unlabeled Turkish corpus used in [17] is not publicly available, in our study we used a different Turkish corpus with size 423M words, called BOUN Web Corpus[4] composed of Turkish news articles and webpage contents. This is the largest Turkish corpus we could obtain and therefore it should be noted that its size, i.e. number of words and tokens in it, could affect the quality and representative coverage of the trained word embeddings with the skip-gram model. Note also that this data set is composed of formal text types in Turkish, but in this study we will test our Turkish NER system on informal text types, such as Turkish microblog texts and more specifically on Turkish tweets. Therefore, we also experiment with word embeddings obtained from large training data composed of unlabeled Turkish tweets. The details of those huge Turkish data sets used in this study to obtain word embeddings will be explained in detail in the next chapter.

3.1.2. Examination of Semantic Relations with Turkish Word Representations

It is shown that word embeddings obtained from a large corpus can capture various interesting linguistic regularities and semantically related words are expected to be close to each other in vector space with distributed representations. The word2vec tool we used in this study allows us to train the model on a large corpus in any language and then query a word to list its closest neighbors in vector space. On our trained model with huge unlabeled data in Turkish, we examined some sample Turkish words and observed that the listed closest words are semantically highly related to the query word. Since our task is NER, we focused on ENAMEX type named entities, namely person, location, and organization names in Turkish while querying words to examine their closest neighbors.

[4] http://79.123.177.209/~hasim/langres/BounWebCorpus.tgz

Table 3.1. Person name sample words and their closest neighbors.

tuğçe *(female name)*	**zeliha** *(female name)*	**musa** *(male name)*	**acar** *(surname)*	**alex** *(soccer player name)*
esra *(female name)*	gülnur *(female name)*	bekir *(male name)*	kızılkan *(surname)*	nobre *(soccer player name)*
tuğba *(female name)*	şerife *(female name)*	cafer *(male name)*	ülger *(surname)*	anelka *(soccer player name)*
zehra *(female name)*	neriman *(female name)*	muhammet *(male name)*	toksoy *(surname)*	appiah *(soccer player name)*
büşra *(female name)*	gülsen *(female name)*	abdurrahman *(male name)*	çavuşoğlu *(surname)*	deivid *(soccer player name)*
aysun *(female name)*	bedriye *(female name)*	mahmut *(male name)*	balkış *(surname)*	luciano *(soccer player name)*
selin *(female name)*	serpil *(female name)*	elyasa *(male name)*	taştop *(surname)*	ailton *(soccer player name)*
didem *(female name)*	necla *(female name)*	seyit *(male name)*	öztürk *(surname)*	marcio *(soccer player name)*

We have shown sample person names as query words on the top row, and below them we have listed the seven closest words in vector space to the input query word in Table 3.1. The top words in the first three columns of this table are Turkish person names and our word embeddings model lists other person names below as their closest neighbors. Note that the query words in the first two columns are female person names in Turkish and their nearest neighbors are also female names, whereas the query word in the third column is a male person name in Turkish and so are its neighbors. The query word in the fourth column is a common surname in Turkish and interestingly, its closest neighbors are also surnames. In the last column, we queried a non-Turkish person name and since it is the name of a famous football player in Turkey, its nearest neighbors are also other famous non-Turkish football player names known well in Turkey. More interestingly, if we compare the first two columns, we may realize that the first query word seems to be a type of modern female person name in Turkish that is used mostly in cities, whereas the second query word seems to be a type of a female person name mostly used in rural areas.

Surprisingly, which might be obvious only for a native Turkish speaker, most of their closest words are also female names that seem to carry this distinction. Another interesting comment to be added here is that, the male person name queried on column three is actually an Arabic name and its nearest neighbors are mostly common Arabic male names used also in Turkey. This examination illustrates that semantically related words are also close to each other in vector space. This feature is very valuable for the NER task since it can help distinguishing named entities based on semantic roles of words. Moreover, traditional language dependent NER models usually benefits from gazetteer lists where common person names are kept as a lookup table to help detecting named entities. These lists are manually constructed for a specific language which is an expensive task, and they can lack surnames and non-Turkish common names where adding also those is an extra burden. However, by employing word embeddings that are obtained with a large Turkish corpus using an unsupervised learning method, we might cover these gazetteer lists and even many more deep semantic relations between words in a specific language without this manual name gathering burden.

Similarly, we have shown location names as query words on the top row, and we have listed below their seven nearest words in vector space in Table 3.2. All the nearest words are also location names as expected. On the first column, we have a country name in Turkish as a query word and all its closest neighbors are also country names in Turkish. Note that since the query country word is a Nordic country name, most of its top neighbors are also Nordic countries. On the second column, we queried a city name in Turkey and observed that its seven closest neighbors are also cities of Turkey, and most of them are even within the same region called Black Sea. On the third and fourth columns, the query words are county names of certain cities and their nearest words are also mostly county or smaller district names within the same city of Turkey. On the fifth and sixth columns, we queried location names as certain districts of Istanbul and observed that the resultant closest words are also district names of Istanbul, which are either geographically or socio-economically close to the query district name. It is quite interesting to see that on column five, when the query word is a district name having a population with a low socio-economic status, most of the closets district names listed also seem to have this property. Similarly on column six, when the query district name is where people with a high socio-economic status live, most of its nearest words seem to be also district names having this status. These examples show how strong word representations can be to capture semantic relations in natural language and how valuable they can be for NLP tasks such as NER. Again, these word embeddings might be a better and cheaper alternative for location names gazetteers used for the NER task, since they are automatically extracted without labeled data and since they can capture many more locational clues, such as a specific port name of a certain city can be very close in vector space to a county name within the same city, which might be hard to capture with generic location names gazetteers.

Table 3.2. Location name sample words and their closest neighbors.

isveç ***(sweden)***	**giresun** ***(city)***	**lapseki** ***(county of çanakkale)***	**kadıköy** ***(county of istanbul)***	**gültepe** ***(district of istanbul)***	**tarabya** ***(district of istanbul)***
norveç *(norway)*	artvin *(city)*	eceabat *(county of çanakkale)*	üsküdar *(county of istanbul)*	çeliktepe *(district of istanbul)*	ortaköy *(district of istanbul)*
hollanda *(netherlands)*	gümüşhane *(city)*	kabatepe *(district of çanakkalel)*	beyoğlu *(county of istanbul)*	nurtepe *(district of istanbul)*	etiler *(district of istanbul)*
belçika *(belgium)*	bayburt *(city)*	geyikli *(district of çanakkalel)*	eminönü *(county of istanbul)*	reşitpaşa *(district of istanbul)*	kireçburnu *(district of istanbul)*
danimarka *(denmark)*	çorum *(city)*	yükyeri *(port of çanakkale)*	bakırköy *(county of istanbul)*	kağıthane *(district of istanbul)*	arnavutköy *(district of istanbul)*
ispanya *(spain)*	bartın *(city)*	gökçeada *(county of çanakkale)*	zeytinburnu *(county of istanbul)*	aydınevler *(district of istanbul)*	emirgan *(district of istanbul)*
portekiz *(portugal)*	sinop *(city)*	bozcaada *(county of çanakkale)*	kağıthane *(county of istanbul)*	güvercintepe *(district of istanbul)*	vaniköy *(district of istanbul)*
polonya *(poland)*	kastamonu *(city)*	gelibolu *(county of çanakkale)*	mecidiyeköy *(county of istanbul)*	harmantepe *(district of istanbul)*	unkapı *(district of istanbul)*

Finally, we have investigated organization names as query words on the top row, and we have listed their seven nearest neighbor words in vector space below in Table 3.3. Our model lists mostly organization entities as closest words to our query words as expected. On the first column of this table, a Japanese technology company name is queried as an organization entity name and we obtained other Japanese technology company names as closest neighbors, plus a South Korean and a Taiwanese one. On the second column, a query word is a mobile phone company name and so are most of its closest words, plus some mobile phone models are listed as related. Note that these specific model names of certain products can be useful to detect organizational named entities. On the third and fourth columns, we investigated automobile manufacturer company names as a query word and observed that similar company names are mostly placed as closest words. When we query with a luxury car company name, its neighboring words also seem to be mostly luxury brands where the top closest ones are also from the same country, and when a regular car company name is queried, we receive mostly other regular car company names as nearest words again where the closest ones are also from the same country. On the fifth column, we have a football club name in Turkish as a query word for organization entity and its closest neighbors are also other football club names in Turkish, where the top nearest ones are the most popular ones. Since football is a very popular topic on Twitter, it is important to detect those club names as organizational entities, which is seen quite often in Turkish Twitter data sets. On the last column, a university name in Turkey is queried as an organization name on our model, which is actually a short name for this university and this type of informal usage is quite common in tweets data sets. We observed that its closest neighbors are also other Turkish university names, again with a short name type of usage. Interestingly, the closest words list also includes the same university name queried but with typos or diacritic variations, or the abbreviation mostly used for this university. Since such typos, diacritics, and abbreviations are very common in informal text types such as Twitter data; placing those kinds of variations of words very close to each other in vector space with word embeddings is a very crucial property where a NER system on informal text types could effectively benefit from. Maybe it will not replace the proper normalization scheme to be applied on informal text types, but if this representative power of word embeddings can be used together even with a simpler normalization scheme, the observed effects of word embeddings can be interesting to be examined.

Table 3.3. Organization name sample words and their closest neighbors.

toshiba *(org.)*	**nokia** *(org.)*	**porsche** *(org.)*	**peugeot** *(org.)*	**galatasaray** ***(football club)***	**boğaziçi** ***(university)***
panasonic *(org.)*	samsung *(org.)*	bmw *(org.)*	citroen *(org.)*	fenerbahçe *(football club)*	bilkent *(university)*
sony *(org.)*	motorola *(org.)*	volkswagen *(org.)*	renault *(org.)*	beşiktaş *(football club)*	bogaziçi *(boğaziçi w/o diacritic)*
sanyo *(org.)*	sagem *(org.)*	audi *(org.)*	bmw *(org.)*	trabzonspor *(football club)*	bü *(abbreviation of boğaziçi)*
fujitsu *(org.)*	w300i *(nokia model)*	volvo *(org.)*	volkswagen *(org.)*	gençlerbirliği *(football club)*	yeditepe *(university)*
hitachi *(org.)*	z530i *(nokia model)*	jaguar *(org.)*	opel *(org.)*	ankaragücü *(football club)*	bahçeşehir *(university)*
samsung *(org.)*	siemens *(org.)*	carrera *(porsche brand)*	ducato *(fiat model)*	bursaspor *(football club)*	odtü *(university)*
benq *(org.)*	w710i *(nokia model)*	maserati *(org.)*	daihatsu *(org.)*	istanbulspor *(football club)*	bağaziçi *(boğaziçi with typo)*

This examination results we presented for closest words in vector space in Table 3.1, Table 3.2 and Table 3.3 show that semantically related words in Turkish are placed close to each other using the unsupervised model we employed. For the NER task, this is a very valuable feature because semantic roles of words can be crucial for recognizing named entities.

3.2. Supervised Stage

The second stage of our neural network based architecture for the NER task on Turkish microblog texts is the supervised stage. At this stage, a comparably smaller amount of labeled data is used for training and constructing the final NER models. The learning algorithm we used and the features we employed to form the NER models at the supervised stage will be explained in detail in the following sections.

3.2.1. The Learning Algorithm

To train our neural network based NER model, we used the implementation presented by Ratinov and Roth [39], which is publicly available[5]. In this work, their implementation of the NER model is based on the regularized averaged multiclass perceptron algorithm, which is presented by Freund and Schapire [40]. This averaged multiclass perceptron algorithm is based on the classical perceptron algorithm proposed by Rosenblatt [41] and applied as a supervised classification algorithm. We will explain the perceptron algorithm briefly below.

Perceptron is a primary computational node in neural networks. Perceptron is a linear classification algorithm that allows for online learning, which means it can process elements in the training set one at a time. It simply makes predictions using the weighted sum of the input feature vector. Here we define a weight vector $w = [w_1, \ldots, w_d]^T$ where d is the number of features we use. The simple perceptron algorithm initializes the weight vector to all zeros, and then the class of a new instance $x_t = [x_1, \ldots, x_d]^T$ is predicted as $\hat{y}_t = sign\ (w^T x_t)$ where $sign(s)$ is 1 if $s > 0$, and -1 otherwise. If this prediction is true, then the weight vector stays the same but if it is false, the weight vector is updated as $w = w + \eta\ (y_t x_t)$ where η is the learning rate. After that, this procedure is applied to the next instance and this perceptron algorithm is run over and over again through the training set until a predefined stopping criteria is met with the currently obtained weight vector. Note that here the classes, or labels, are defined to be in {-1, +1} and hence the classical perceptron algorithm is a binary classification algorithm.

This simple perceptron algorithm can be easily applied to multiclass classification, as shown in [42]. With this linear online multiclass version of the perceptron algorithm, a prediction out of k classes is made for a d-dimensional instance vector x_t as an input at each round t. Note that this online learning is performed in a sequence of consecutive

[5] http://cogcomp.cs.illinois.edu/Data/ACL2010_NER_Experiments.php

rounds. The goal is defined to minimize the number of prediction mistakes M on each run over the training set, which is defined as

$$M = \sum_{t=1}^{T} \mathbf{1}[\hat{y}_t \neq y_t] \tag{3.4}$$

where T is the total number of rounds at each run over the training set, or simply T is the total number of instances in the training set. Here, the predicted class $\hat{y}$ is defined as

$$\hat{y} = \underset{j \in [k]}{\operatorname{argmax}} \ (Wx)_j \tag{3.5}$$

where $\mathbf{1}[\pi]$ is 1 if predicate π is true and 0 otherwise, and $(Wx)_j$ is the j^{th} element of the vector obtained by multiplying the weight matrix W, $W \in \mathbb{R}^{kxd}$, with the instance vector x. Again, this weight matrix is initialized to all zeros at the beginning of the multiclass perceptron algorithm. At the end of each round, based on whether the weight matrix predicts the current label correctly or not, the algorithm updates its weight matrix with parameters in order to predict the future instances better. If the label is predicted correctly, then the weight matrix is not updated at all. On the other hand, if a false label is predicted, the weight matrix is updated such that after each round t, x_t is added to the y_t^{th} row and subtracted from $\hat{y}_t^{th}$ row of the weight matrix W. To be more precise, the weight matrix is updated as follows

$$W^{t+1} = W^t + \eta U^t \tag{3.6}$$

where $W^1 = 0$ at the initial step and where the matrix U^t, $U^t \epsilon \ \mathbb{R}^{kxd}$, is defined as

$$U_{r,j}^t = x_{t,j}(1[y_t = r] - 1[\hat{y}_t = r]) \tag{3.7}$$

At the end of training, the final weight vector is stored and it is used to predict the class, or label, using equation 3.5, for each instance in the test set. In Figure 3.2, we present an illustration of the multiclass perceptron network, which is simply a two-layer neural network composed of an input layer and an output layer. Note that here, a d-

dimensional input vector x_i, where $i = 1, \ldots, d$, is assigned to one output class y_j , where $j = 1, \ldots, k$, out of k possible classes. W_{ij} represents the weight of the arrow from input x_i to output y_j. With this illustration, the input values are multiplied with the corresponding weight values and summed up to construct the output values. The predicted class or label will be based on the maximum of these resulting output values.

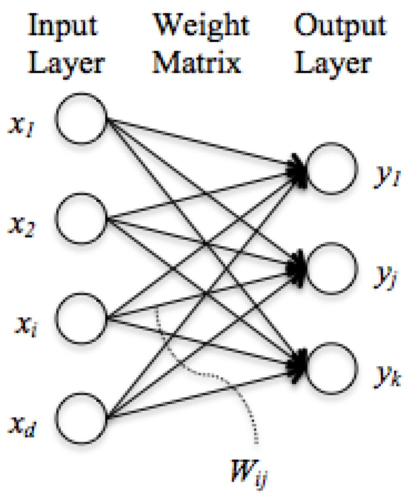

Figure 3.2. Multiclass perceptron as a simple two-layer neural network structure.

Note that in this classical perceptron algorithm, only the final weight vector values are kept as the learnt parameters and used for the test instances. However, Freud and Schapire [40] proposed that it would be better if we can store also the previously estimated weight matrix values. The idea stems from the fact that a previously achieved and stable weight matrix can be ruined with updates at the last instances of the training set.

To achieve improved generalization by employing the previous weight matrix parameters, modifications to the standard perceptron algorithm are proposed in [40]. One of these modifications is called the averaged perceptron approach that we also used in our study. This approach is actually very similar to the classical multiclass perceptron algorithm; but in the averaged multiclass perceptron algorithm, we keep track of the average weight matrix, $W_{avg} \in \mathbb{R}^{kxd}$, as well. The average weight matrix is initialized to all zeros at the beginning and at each iteration, the current weight matrix is added to W_{avg}. When the training is over, this matrix is divided by the total number of iterations during the training and hence, the average of all weight matrices is obtained. With this new approach, instead of the final weight matrix values, we simply return the average weight matrix W_{avg} values and use these parameters for the test instances. This averaged multiclass perceptron algorithm is also shown in Figure 3.3, which is taken from [16].

```
Data: Training data $T = \{(x_t, y_t)\}_{t=1}^{|T|}$, learning rate $\eta$ and number of epochs
      $N$
Result: Learned averaged weight matrix $W_{avg}$
$W \Leftarrow 0; W_{avg} \Leftarrow 0;$
for $n = 1$ to $N$ do
    for $t = 1$ to $T$ do
        $\hat{y} \Leftarrow \arg\max_j (Wx_t)_j;$
        if $\hat{y} \neq y_t$ then
            $W_{y_t} \Leftarrow W_{y_t} + \eta x_t;$
            $W_{\hat{y}} \Leftarrow W_{\hat{y}} - \eta x_t;$
        end if
        $W_{avg} \Leftarrow W_{avg} + W;$
    end for
end for
$W_{avg} \Leftarrow W_{avg}/(N \times T);$
return $W_{avg};$
```

Figure 3.3. Averaged multiclass perceptron algorithm [16].

In this algorithm, W_r represents the r^{th} row of W. In our study, we chose the learning rate η to be 0.1. Moreover, we used the stopping criteria during the training of our model such that if the performance does not improve for the last 10 epochs, the training stops. Here, epoch represents each run over the training set. The performance we used for checking the stopping criteria is the one that we measure after each epoch over the whole training set. The epoch that performed best is then chosen to be our final model.

3.2.2. The General Framework

In this section, we will describe the features we used to train our NER model and the representation of the named entities we employed will be explored together with the other design choices we made to complete our framework.

In our study, the framework we used is similar to the one presented by Ratinov and Roth [39] and also employed in [17] for Turkish NER. As done in [39] and [17], we explored both local and non-local features to construct our Turkish NER model, but we did not finally use non-local features since they have no benefit for the domain of short Twitter messages. The features we examined for our NER system are listed as follows:

- *Local features*: These are the features mostly related to the previous and next tokens of the current token x_i. The local features we exploited are as follows:
  - (i) *Context*: All tokens in the current window with size two, i.e. $c_i = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
  - (ii) *Capitalization*: A Boolean feature indicating whether the first letter of a token is upper-case or not. This feature is generated for all the tokens in the current window c_i
  - (iii) *Word type information*: Type information of tokens in the current window c_i, i.e. all-digits, contains-apostrophe, all-capitalized and all-non-letters
  - (iv) *Token prefixes*: First characters with length three and four, if exists, of current token x_i
  - (v) *Token suffixes*: Last characters with length one to four, if exists, of current token x_i
  - (vi) *Previous tags*: Named entity tag predictions of the previous two tokens, i.e. y_{i-1}, y_{i-2}
  - (vii) *Word embeddings*: d-dimensional vector representations of the words in the current window c_i, i.e. $(e_{i-2}, e_{i-1}, e_i, e_{i+1}, e_{i+2})$
- *Non-local features*: These are the features where global and hence non-local dependencies are taken into account. The non-local features we exploited are as follows:

(i) *Context aggregation*: When tokens that are identical to the current token appear in several location in the text, in addition to the context features of the current token, we aggregate the context features of all such tokens within a window of size 200, as proposed in [39].

(ii) *Extended prediction history*: When making a prediction for the current token, the tag assignment distribution for all tokens in the previous 1000 words that are identical to the current token is recorded, as proposed in [39]. This tag distribution of the current token is then used as an extended prediction history feature.

Note that although non-local features are proven to be useful for the NER task on long and formal text types such as news articles, their usage and benefit is questionable for informal and short text types. In Twitter, the text size is limited to 140 characters for each tweet. This limitation brings an additional burden on the NER task for tweets. Due to the fact that each tweet is treated as a single document with only 140 characters, it is difficult to make use of non-local features such as context aggregation and prediction history for the NER task on tweets, whereas these features are generally used for formal text types mainly composed of news articles. With this motivation, firstly we have checked the benefit of these non-local features with our Turkish NER system tested on Turkish tweets, and observed that we achieve better results without those non-local features, which actually makes sense. Therefore, throughout this study, we decided not to use non-local features for our NER models designed to work on Turkish tweets. We have also tested our models, which use word embeddings, with and without the capitalization feature on tweets and we have seen that not using the capitalization feature is better for tweets without normalization, whereas using the capitalization feature is better if we apply a normalization scheme on Turkish tweets first. We will present the details of our findings about the best set of those features for the NER task on Turkish Twitter data in the experiments and results chapter.

Another important point about those explored features is that, since we have trained our Turkish NER models on annotated Turkish news data instead of Turkish tweets data, due to the lack of enough amount of annotated Turkish Twitter data, we could not explore Twitter specific features much at this stage. However, our NER system still has Twitter specific parts such that during the Turkish text normalization stage on tweets before we apply NER on them, which will be explained in detail in the next section, we have extracted Twitter specific keywords for certain Twitter text usages such as mentions, hashtags, smileys and URLs. In addition, we have also extracted Twitter specific keywords on our large Turkish tweets corpus during the Twitter processing stage, which will be explained in detail later in the data sets chapter, and again this makes our Turkish NER system Twitter specific. Moreover, since we have trained our Turkish word embeddings by using this additional corpus composed of Turkish tweets, and since we used word embeddings obtained from those processed tweets corpus as a crucial feature for our NER model, our word embeddings feature can be thought as a Twitter specific feature for our NER model.

The representation scheme for named entities is significantly important in terms of performance for a NER system. The two most popular such encoding schemes are BIO and BILOU. The BIO scheme identifies the **B**eginning, the **I**nside and the **O**utside of the text segments, which are named entities in our case. On the other hand, the BILOU representation scheme identifies the **B**eginning, the **I**nside and the **L**ast tokens of multi-token named entities, plus the **O**utside if it is not a named entity and the **U**nit length if the entity has single token. Obviously, BILOU scheme allows a more expressive model to be learnt compared to the BIO scheme. Since it is shown in [39] that BILOU representation scheme significantly outperforms the mostly used BIO encoding scheme, we make use of the BILOU encoding scheme for tagging named entities in our study. After we tag the named entities using the BILOU encoding scheme, the tags are converted to BIO tags while testing since this was necessary to use the standard CoNLL performance evaluation. During this conversion, the tags U and L are simply converted into B and I, respectively.

In our model, the feature vectors are concatenated and then given to the averaged multiclass perceptron as shown in Figure 3.2 in order to predict the named entity tag of a given word. At this stage, we have used 1-of-N encoding for discrete variables with three or more alternative values. For instance, if we have only four possible values for a type of a feature, their encoding will be as follows: 1000, 0100, 0010 and 0001. On the other hand, since the system can observe an unknown feature during testing that has not been observed in the training, an encoding for unknown features is necessary too. As a result, the final encoding in our example will be as follows: 10000, 01000, 00100, 00010 and 00001, where the first four encodings are for the known features and the last encoding is for unknowns. Note also that we normalized the numbers as done in [39]. As an example, 2015 is represented as *DDDD* and (0212) 123 45 67 is represented as (*DDDD*) *DDD* *DD* *DD*. This type of numeric normalization allows to achieve a degree of abstraction for numeric expressions such as phone numbers, years etc.

The experiments we performed with different models having the variations of this general framework and the detailed comparison of their results for the NER task on Turkish Twitter data will be presented in the experiments and results chapter of this study. We will explain these model parameters, where most of them convert our generic model into a microblog texts specific model, in detail in the following section.

3.3. Model Parameters

In this section, we will mention about the differences between the various NER models we experimented with and evaluated on Turkish Twitter data. We have a list of model design options or experimental setting options that we can manipulate in order to see their effects on the NER task for Turkish tweets. These are namely the types of annotated texts we used for training, the variations on the annotated Twitter test sets, the unlabeled source text type of word embeddings, using the capitalization feature or not, and finally applying normalization on tweets data sets.

3.3.1. Training on Different Text Types

Since our task is recognizing named entities on Turkish Twitter messages, which are informal text types and hence very different in nature from formal text types such as news articles, a successful NER model for Turkish tweets ideally should be trained on labeled data composed of similar informal text types, i.e. Turkish tweets. On the other hand, the amount of annotated data used for training a NER system highly effects its generalization ability and hence the performance on unknown test data in the future.

Since we lack a large amount of Turkish Twitter data annotated with named entities, our first attempt is to train a NER model on a large amount of annotated Turkish news data, similar to what has been done in previous works on Turkish Twitter NER, presented in [9] and [29]. Although we have a limited amount of annotated Turkish tweets data, our second attempt is to train a NER model on this relatively small amount of tweets data with 10-fold cross-validation and compare its results with our first attempt.

3.3.2. Testing on Different Twitter Data Sets

Our Turkish NER system will always be tested on microblog texts data, i.e. Turkish Twitter data sets annotated with named entities. We have two different data sets for that purpose, which will be explained in detail later in the data sets chapter. These are namely TwitterDS-1 composed of around 5K tweets and TwitterDS-2 composed of around 2.3K tweets, presented in [29] and [9], respectively.

In TwitterDS-1, we observed some serious problems that can affect the performance of our final NER system. The first problem we observed is that, it is not annotated properly such that a lot of named entities that should be annotated are missed and hence remained as not annotated. For this problem, we have no solution since it requires manual check on all 5K tweets. Nevertheless, we can still use this data set as it is for comparison with the previously proposed NER models on Turkish tweets. However, as we will see in the experiments and result chapter, the results reported on this data set are comparably worse than the results reported on TwitterDS-2, which might have been caused by this missed annotations.

The second problem we observed in TwitterDS-1 is that, as also discussed in [9], 53 named entities annotated as location names are actually referring to organization names with metonymic readings. When we examined these entities, we observed that in all of them, football club names are actually referred with their home city names. Such examples are harder to be recognized correctly by a NER system since contextual clues are highly required together with semantic relations, which are limited in short text messages such as tweets. Therefore, if we leave such entities with location tags, the performance will be better compared to when we correct them with organization tags. For the sake of measuring the actual performance, we have corrected these location names as organization names and we will call this updated version of the first Twitter data set as TwitterDS-1_LTOC where LTOC stands for location-to-organization-corrected for such cases. We will use both TwitterDS-1 and TwitterDS-1_LTOC data sets as test sets in all of our experiments.

The third problem we observed in TwitterDS-1 is that it includes some non-Turkish tweets. To be more precise, we realized that 291 tweets out of 5040 are non-Turkish in TwitterDS-1, which may affect the performance of our NER system designed for Turkish, i.e. trained on labeled Turkish data and using Turkish word embeddings. By non-Turkish tweets we mean the messages are composed of completely non-Turkish words, not the ones having both Turkish and non-Turkish words together, since it is quite common to use some non-Turkish words such as foreign names or lyrics etc. in Turkish tweets. As an improvement, we manually filtered out those non-Turkish tweets both from TwitterDS-1 and TwitterDS-1_LTOC and renamed them as TwitterDS-1_FT and TwitterDS-1_LTOC_FT respectively, where FT stands for fully Turkish. In addition to TwitterDS-1 and TwitterDS-1_LTOC data sets, we will also use both TwitterDS-1_FT and TwitterDS-1_LTOC_FT data sets as test sets in all of our experiments to observe their effects on performance.

On TwitterDS-2, we have not observed the above problems and therefore we leave it as it is, as a much healthier annotated test set for our Turkish NER system. In [9], it is noted that non-Turkish tweets have already been manually filtered out from TwitterDS-2, which is consistent with what we have observed. All in all, we mainly have two different annotated Turkish Twitter test sets, but since we created three additional ones with corrections from the first set, we will have five different test sets in total to be used in all of our experiments.

3.3.3. Using Word Embeddings from Different Text Types

Since our task is NER on informal text types such as Turkish tweets, not on formal text types such as Turkish news, the type of texts found in the unlabeled Turkish corpus used to attain the word embeddings can affect the performance results. With this intuition, in addition to learning Turkish word embedding from the BOUN Web Corpus mostly containing formal text types such as news articles, we also experimented with the word embeddings attained from an unlabeled Turkish corpus composed of only informal text types, i.e. tweets, at the unsupervised stage. As a third option, we will compare the effects of word embeddings obtained from the combination of these two Turkish corpora, large amount of unlabeled Turkish news corpus plus unlabeled Turkish tweets corpus, which can be interesting to examine for the Turkish NER task on tweets.

With this motivation, we trained basically three different models for Turkish NER task on Twitter. The first model uses only the word embedding obtained from the BOUN Web Corpus, where we will explain its details in the data sets chapter. The second model uses only the word embeddings obtained from our tweets corpus, which has nearly half of the size compared to the BOUN Web Corpus, where its details will be examined again in the next chapter. The third and the final model uses a combined corpus composed of both the BOUN Web Corpus and tweets corpus in order to attain word embeddings, which will be used as an important feature during the training of our NER system at the supervised stage. We will report and compare the results of all these three basic models in our experiments.

3.3.4. Capitalization

In our general NER model framework, we mentioned about the capitalization feature as a local feature, which basically takes into account the capitalization of tokens in the current window. It is an important feature especially for formal text types such as Turkish news data, where proper capitalization of named entities is widely seen. However, we realized that proper capitalizations of named entities are often missed in informal text types such as Turkish tweets, due to their unstructured nature and everyday language type of usage. With this motivation, we evaluated our Turkish NER models firstly trained by using the capitalization feature and then also trained by not using the capitalization feature in order to compare its effects. In the second case, where we relaxed this capitalization, our Turkish NER models may consider all tokens as possible named entity candidates without checking whether their initial letters are capitalized or not, so that capitalization is not a strong clue for named entities anymore. We will report and compare the performance results of both types of models in our experiments.

3.3.5. Normalization

The nature of real data, especially for microblog texts such as tweets, is highly noisy which creates a great burden and challenge when performing NLP tasks on them. We mentioned about those Twitter NLP challenges in Section 1.3. In order to neutralize the issues of writing and spelling errors, such as punctuation violations with lack of proper capitalization or apostrophes, usage of non-diacritic characters especially in Turkish tweets, abbreviations, slangs and special modifications found in tweets, it is highly recommended to apply a normalization scheme to correct and expand such usages before using the NER system on tweets.

With this motivation, we experimented on both non-normalized and normalized Turkish tweet sets in order to see the effect of tweet normalization as a preprocessing step for the Turkish NER task on tweets, especially with our proposed models using word embeddings. For that purpose, we have used the web-based Turkish normalization interface[6] presented as a component of ITU Turkish Natural Language Processing Pipeline[7], which we will call as ITU Normalizer in our study. This normalizer tool is publicly available for research purpose and the details of this work are presented in a recent publication by Torunoğlu and Eryiğit [43]. It is the first study on Turkish text normalization that is publicly available where the model is designed and tested especially for Turkish social media text. This model has mainly two stages as ill formed word detection and candidate word generation. The second part contains seven stages, namely letter case transformer, replacement rules and lexicon lookup, proper noun detector, deasciificator that corrects non-diacritics, vowel restoration, accent normalizer and spelling corrector where all of those runs sequentially. During this normalization, at the replacement rules stage they also labeled Twitter-specific words and usages such as mentions, hashtags, smileys, vocatives and Twitter-keywords like RT for retweets etc.

[6] http://tools.nlp.itu.edu.tr/Normalization
[7] http://tools.nlp.itu.edu.tr/

In addition to this normalization scheme, we also applied our own post processing steps such that we replaced the Twitter specific keywords given by ITU Normalizer into our own format of Twitter related keywords such as TwitterMention, TwitterHashtag, TwitterSmiley, TwitterUrl etc. We performed this process on top since we already applied this type of Twitter keyword tagging scheme on our huge unlabeled Twitter corpus where we obtained our Turkish word embedding from, as we will explain in Section 4.1.2 and we call this as Twitter processing. In order to have consistency between Twitter specific keywords, we believe that this step was necessary and observed that it improves the performance when our tweets corpus is used for training the word embeddings in our NER models.

Another additional processing we applied is keeping the PLO tagging format texts, such as “[PER person_name]”, after this normalization. We realized that if we apply normalization on a Twitter data set annotated with named entities, which includes certain PLO tagging texts, then we lose some of our PLO tagger texts with correct format, such that “[PER oguz]” becomes “pera Oğuz]” after this normalization. The results of normalization also vary with those additional PLO tag texts, which are meaningless and out-of-vocabulary words for Turkish. In order to avoid this problem, we firstly removed all PLO tagger keywords together with their square brackets from the annotated tweets data and obtained a raw tweets version of it. Then we normalized this raw data and then added those PLO tagging structure in order to obtain an annotated and properly normalized Turkish Twitter data set.

We applied this normalization method on all five of our annotated Twitter data sets with variations as explained in Section 3.3.2. As a result, we obtained five additional normalized Twitter data sets annotated with named entities, namely TwitterDS-1_Norm, TwitterDS-1_LTOC_Norm, TwitterDS-1_FT_Norm, TwitterDS-1_LTOC_FT_Norm and TwitterDS-2_Norm. We will report the performance results on all of those five normalized data sets and compare them with the results obtained on the not normalized version of each in our experiments.

4. DATA SETS

In this chapter, we will present the details of the data sets we have used for training and testing of our NER system. In general, we have used two groups of Turkish data sets. The first group is the unlabeled data set that we have used in the unsupervised stage to obtain Turkish word embeddings and the second group is the labeled, i.e. annotated with named entities, data set that we have used in the supervised stage for training and testing.

4.1. Unlabeled Data Sets

In the unsupervised stage, we used two different groups of large unlabeled data sets, or corpora. The first group of data is composed of huge amount of formal texts that are collected from Turkish news articles and web documents with no annotations for training purposes of our Turkish word embeddings at the unsupervised stage. The second group of data is composed of informal text types that are collected from a large amount of Turkish Twitter messages with no annotations.

4.1.1. Turkish News Corpus

We used the Turkish corpus[8] prepared by Sak *et al.* [44] in the unsupervised stage in order to learn Turkish word embeddings. It is a large text corpus for Turkish that is compiled from the web and composed of four sub corpora. Three of them are collected from three popular Turkish newspapers, called as “NewsCor”, and the other one is collected by general sampling of Turkish web pages, called as “GenCor”. The combination of these two corpora is called “BOUN Corpus”. This corpus is further cleaned after processing by using certain heuristics and a morphological parser. You can find the details of this work in [44]. We further tokenized the data using the open source Zemberek[9] tool developed for Turkish and also applied lowercasing. The goal of lowercasing this corpus is to limit the number of unique words.

[8] A text corpus compiled from the web: http://79.123.177.209/~hasim/langres/BounWebCorpus.tgz
[9] https://github.com/ahmetaa/zemberek-nlp

Table 4.1. BOUN web corpus size statistics taken from [44].

Corpus	Words	Tokens	Types
Milliyet	59 M	68 M	1.1 M
Ntvmsnbc	75 M	86 M	1.2 M
Radikal	50 M	58 M	1.0 M
NewsCor	184 M	212 M	2.2 M
GenCor	239 M	279 M	3.0 M
BOUN Corpus	423 M	491 M	4.1 M

The unlabeled data set, namely the BOUN Web Corpus, which we used in the unsupervised stage, contains around 423M words and around 491M tokens in total. The total number of types, or the vocabulary size, in the corpus is reported as 4.1M. The detailed statistical information, taken from [44], about the number of words (i.e. all words found in the corpus), number of tokens (i.e. all words plus lexical units like punctuation marks) and number of types (i.e. distinct tokens, or vocabulary size) of all four sub corpora grouped under "NewsCor" and "GenCor" within this large Turkish corpus is presented in Table 4.1.

We benefit from this huge unlabeled data in the unsupervised stage for obtaining the continuous space vector representations of words in Turkish. Our ability to train our model with large amount of unlabeled data is notable because as the amount of data increases, the representativeness of the feature vectors of the words that are obtained from the corpus increases as well. This situation also means that these highly representative feature vectors of words will be more useful in the supervised stage where we will use these word representations as additional features for our supervised stage of our NER system.

4.1.2. Turkish Tweets Corpus

Note that our task is recognizing named entities on informal text types such as Turkish Twitter data, not on formal text types such as Turkish news media data. Therefore, the type of texts found in the huge amount of unlabeled Turkish data set used to obtain word embeddings at the unsupervised stage might be important. Based on this intuition, in addition to the BOUN Web Corpus composed of mostly formal text types, we also explored a large unlabeled Turkish corpus composed of only informal text types, i.e. short Twitter messages called tweets. We thought that comparing the effects of word embeddings obtained from a huge amount of unlabeled Turkish tweets corpus in addition to the Turkish news corpus might be interesting for the NER task on Turkish Twitter data. In order to construct a large unlabeled Turkish tweets data set, we have used two different tweets corpora in Turkish.

The first unlabeled Turkish tweets corpus we used is called TS TweetS corpus[10] that is composed of around 1M Turkish tweets with around 13M words, prepared by Taner Sezer [45] and released recently in February 2014. TS TweetS is one of the four existing corpora located under the TS Corpus project and is publicly available at TS Corpus server[11]. Note that although TS TweetS corpus originally includes specific tags for morphological analysis, POS and other special tags for Internet language, we only used the untagged version of this 1M tweets corpus in order to obtain word embeddings from unlabeled Turkish corpus.

The second unlabeled Turkish tweets corpus we used is called 20 million Turkish Tweets[12], which is composed of around 20M tweets, prepared by Bolat and Amasyalı from Kemik[13] NLP Group of Yıldız Technical University. This raw Turkish tweets data set is publicly available and we will refer to this corpus as Kemik Tweets. Note that at each line of this raw data set, we have the detailed date and time information of the tweet, the username of the owner of this tweet and finally the text body of the tweet. We processed this raw data set in order to obtain only text bodies of tweets and after that, the size of this processed Turkish tweets corpus is measured to be having around 228M words in total.

In order to construct a huge amount of unlabeled Turkish tweets corpus, we combined these two tweets corpora and we obtained around 21M Turkish tweets in total. As a preprocessing, firstly we processed this combined large raw tweets corpus in order to

[10] http://tscorpus.com/en
[11] http://gui.tscorpus.com/
[12] http://www.kemik.yildiz.edu.tr/data/File/20milyontweet.rar
[13] http://www.kemik.yildiz.edu.tr/?language=en

extract only text bodies of tweets. After that, since this data set is composed of Twitter specific texts, we applied what we call Twitter processing where we replaced "@<username>"s with TwitterMention, "#<hashtag>"s with TwitterHashtag, smileys with TwitterSmiley and URLs with TwitterUrl keywords. After that, we applied tokenization on this tweets corpus using again the publicly available Zemberek tool designed for Turkish. Finally, we applied lowercasing on this unlabeled Turkish tweets corpus just like we did on Turkish news corpus, with the aim of limiting the number of unique words. After these preprocessing steps, using the word2vec tool, we obtained Turkish word embeddings using this large amount of combined and preprocessed Turkish tweets corpus.

The purpose of this Twitter processing is to limit the number of unique words to be represented in the vector space. To be more precise, each username on Twitter is often a unique composition and usually not a common word or phrase in natural language, and so are the Twitter specific hashtags which are composed of multiple words with no space. Moreover, URL texts are also unique compositions of characters with specific format, which usually have no meaning in natural language, similar to what we have with the smileys. Replacing such Twitter specific or internet specific sequence of characters with fixed keywords allows us to decrease the sparsity in vector representations of words obtained from Turkish Twitter corpus, and enables us to achieve a degree of abstraction which in turn hopefully increases the representative power of the obtained Turkish word embeddings. In addition, this Twitter processing prevents us from problems with tokenization on tweets corpus, since we observed that unnecessarily increased number of tokens are extracted, which usually have no meaning in natural language, especially with URLs and smileys when we leave them as they are.

The data size of this combined and preprocessed Turkish tweets corpus, in terms of number of words and number of tokens, is summarized in Table 4.2 together with the sizes of the two different tweets corpora that we combined to obtain the Turkish Twitter corpus. You can also observe the effects of the preprocessing steps on this combined raw tweets corpus by examining the number of words and tokens obtained from the last three rows of Table 4.2. Note that since TweetS corpus was already tokenized, the number of words and tokens in tweet bodies are found to be the same after processing, i.e. after getting tweet text bodies only, and even after Twitter processing, as expected. Therefore, we benefit from Twitter processing step mainly on the other corpus, which is much larger than the first one.

Table 4.2. Turkish tweets corpus size statistics.

Corpus	Tweets	Words	Tokens
TS TweetS (Raw)	1 M	15 M	20 M
Kemik Tweets (Raw)	20 M	398 M	579 M
Tweets Corpus (Raw)	21 M	413 M	599 M
TS TweetS (Processed)	1 M	13 M	13 M
Kemik Tweets (Processed)	20 M	228 M	314 M
Tweets Corpus (Processed)	21 M	241 M	327 M
TS TweetS (Twitter Processed)	1 M	13 M	13 M
Kemik Tweets (Twitter Processed)	20 M	228 M	280 M
Tweets Corpus (Twitter Processed)	21 M	241 M	293 M

After processing the combined raw corpus in order to get tweet bodies only and then applying Twitter processing to replace Twitter specific texts with predefined keywords, we finally have around 241M words and around 293M tokens in total in this large corpus of Turkish tweets. This is the largest Turkish tweets corpus we could obtain publicly and its size is comparably less than the Turkish news corpus composed of around 423M words and around 491M tokens.

4.2. Labeled Data Sets

In the supervised stage, we used two different groups of labeled data sets for training and testing purposes of our Turkish Twitter NER system. The first group of data is composed of formal text types that are collected from Turkish news articles and annotated with named entities for training purposes of our NER system. The second group of data is composed of informal text types that are collected from Turkish Twitter messages and annotated with named entities for testing purposes of our NER system. The details of these two different groups of annotated Turkish data sets can be found in the following sections, together with their resource information.

4.2.1. Turkish News Data Set

The labeled Turkish news data set, which is used for training purposes during the supervised stage, is the one prepared by Tür *et al.* [11]. This data set is commonly used for performance evaluation of NER systems for Turkish, including the ones presented in [6], [15] and [17]. It is based on the general news domain that is collected from online Turkish newspaper articles and therefore it is composed of formal text types, mostly with proper spelling and grammar. In total, this data set is composed of around 500K words. Yeniterzi [15] partitioned this data set into training and test sets such that one tenth of the data with around 50K words is reserved for testing and the rest with around 450K words is used for training purposes. The whole data set is annotated with respect to the ENAMEX types, or PLOs only. In total, it includes around 24K person names, 14K location names, and 16K organization names. You can find the number of words and number of named entities, as presented in [15], after the training and test sets partitioning in Table 4.3.

Table 4.3. Number of words and named entities in the labeled Turkish news data set [15].

	Train	**Test**
Data Size (#words)	445,498	47,344
Person	21,701	2,400
Location	12,138	1,402
Organization	14,510	1,595
Total PLOs	48,349	5,397

Note that we ignored the test partition and only used the training partition of this labeled Turkish news data set for the training purposes of our NER system. The test partition of this data set is used as a validation set during the training phase of our NER system. Another important point to state here is that, these numbers of named entities presented in Table 4.3 are all based on token-level. If we count the named entities composed of multiple tokens as only one entity, then these numbers will drop to 14,481 person names, 9,409 location names and 9,034 organization names in the training partition, as presented in [17], with a total of 32,924 PLOs in the training partition which we used for training purposes. Since the CoNLL evaluation task takes into account only phrases and not tokens, this last counting mechanism can be much more meaningful to visualize the size of the data set in terms of named entities.

4.2.2. Turkish Twitter Data Sets

The labeled Turkish Twitter data sets, which are used for testing purposes during the supervised stage, are composed of two different data sets. Both of them are composed of Twitter messages, or tweets collected from the most popular microblogging site Twitter. Therefore, they both contain informal text types that lack proper spelling and grammar rules and their nature is different from formal texts such as the ones from the news domain.

The first labeled Twitter data set, which we call TwitterDS-1, is composed of Turkish tweets prepared by Çelikkaya *et al.* [29]. It is annotated manually based on the guidelines of the MUC-6 NER task. To the best of our knowledge, it is the first data set collected and annotated especially for the Turkish Twitter NER task. It contains ENAMEX, NUMEX, and TIMEX entity types. We will use only the ENAMEX type named entities in our supervised stage. The size of this data set is given as around 50K tokens. It contains 1,336 PLO tokens in total. The exact number of tokens for each type of named entities, as presented in [29], is given in Table 4.4.

The second labeled Twitter data set, which we call TwitterDS-2, is composed of Turkish tweets prepared by Küçük *et al.* [9]. It is composed of 2,320 tweets after the cleaning process of filtering the non-Turkish tweets and retweets. The data size is reported as around 21K tokens. This data set is also annotated manually following the NER guidelines of MUC. It contains ENAMEX, NUMEX and TIMEX entity types, plus a MISC type which is defined for names of TV programs, movies, music bands, products and other named entity types found in the data set. Note that we will only use its ENAMEX type named entities in our supervised stage. This data set contains 980 PLO phrases in total. The exact number of annotated named entities and their types, as presented in [9], are given in Table 4.5. Note that this annotated Turkish Twitter data set is publicly available[14] with the Tweed ID's and their corresponding named entity tags.

Table 4.4. Number of tokens for each named entity type in the annotated Turkish Twitter data set: TwitterDS-1 [29].

TwitterDS-1	
Data Size (#tokens)	54,283
Person	676
Location	241
Organization	419
Total PLOs	1,336
Date	60
Time	23
Money	14
Percentage	4
Total Basic Seven Types	1,437

Table 4.5. Number of named entities for each named entity type in the annotated Turkish Twitter data set: TwitterDS-2 [9].

TwitterDS-2	
Data Size (#words)	20,752
Person	457
Location	282
Organization	241
Total PLOs	980
Date	201
Time	5
Money	16
Percentage	9
Total Basic Seven Types	1,211
MISC (others)	111
Total Annotated Types	1,322

[14] http://optima.jrc.it/Resources/ 2014_JRC_Twitter_TR_NER-dataset.zip

5. EXPERIMENTS AND RESULTS

In this chapter, we will present the details of various experimental setups and the performance evaluation of our different models. We will also compare the results of our NER models with the state-of-the-art systems developed for NER in Turkish tweets.

We have a list of experimental setting options or model design options that we can alter in order to observe their effects on the Turkish NER task for tweets. These are the types of annotated texts we used for training, the variations on the annotated Twitter test sets, the unlabeled source text type of word embeddings, using the capitalization feature or not, and finally applying normalization on tweets data sets. In Section 3.3, we have explained the differences between our various NER models that we have experimented with. In this chapter, we will present the results of each such system and compare their performance results on Turkish Twitter data.

5.1. Experiments with NER Models Trained on News Data

Throughout this section, we will present the performance results of our various Turkish NER models trained on annotated Turkish news data prepared by Tür *et al.* [11].

5.1.1. Using Capitalization as a Feature

In this subsection, we will present both the phrase-level and token-level performance results of our three different Turkish NER models using the capitalization feature as a clue for recognizing named entities. These are the test results obtained on the five different annotated Turkish Twitter data sets that we described in Section 3.3.2.

In the first NER model, we used word embeddings obtained from Turkish web corpus, called BOUN Web Corpus, and we name this model as NER Model Web. In the second NER model, we used word embeddings obtained from Turkish tweets corpus and we name this model as NER Model Tweets. The third and also the final model in terms of word embeddings source is constructed by using word embedding learnt from the composition of these two corpora, and we name this model as NER Model Web+Tweets.

Note that all of these Turkish NER models' results on all five tweets data sets are evaluated with respect to the CoNLL metric and shown in the phrase-level results parts of Table 5.1, Table 5.2 and Table 5.3 for NER Model Web, NER Model Tweets and NER Model Web+Tweets, respectively. We also reported token-level results, which are generally higher than phrase-level results as expected, since we believe that partially correct predictions can also be valuable in the context of tweets for various applications such as filtering or fetching tweets with named entities.

Table 5.1. Phrase-level and token-level F-score performance results of NER Model Web on Turkish Twitter test sets, using the capitalization feature.

NER Model Web	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	31.36	44.82	31.37	33.68	39.33	45.07	31.10	38.12
TwitterDS-1	31.36	51.98	35.17	36.55	39.33	51.22	34.44	40.42
TwitterDS-1_LTOC_FT	37.06	45.73	33.00	37.37	46.59	45.85	32.68	42.51
TwitterDS-1_FT	37.06	52.87	37.05	**40.53**	46.59	51.98	36.26	**45.06**
TwitterDS-2	49.86	66.39	41.83	**53.14**	62.46	65.15	50.20	**60.23**

Table 5.2. Phrase-level and token-level F-score performance results of NER Model Tweets on Turkish Twitter test sets, using the capitalization feature.

NER Model Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	31.26	41.33	30.18	32.51	37.68	42.70	31.14	36.60
TwitterDS-1	31.26	49.01	33.21	35.14	37.68	49.15	33.93	38.69
TwitterDS-1_LTOC_FT	37.28	42.71	34.67	37.42	45.34	43.43	35.37	42.20
TwitterDS-1_FT	37.28	50.29	38.72	**40.44**	45.34	49.88	39.02	**44.61**
TwitterDS-2	42.78	65.42	32.84	**47.72**	58.02	64.87	44.09	**56.68**

Table 5.3. Phrase-level and token-level F-score performance results of NER Model Web+Tweets on Turkish Twitter test sets, using the capitalization feature.

NER Model Web+Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	35.03	41.75	32.11	35.30	42.59	41.60	31.67	39.57
TwitterDS-1	35.03	49.43	35.93	38.11	42.59	48.28	34.98	41.83
TwitterDS-1_LTOC_FT	41.08	42.47	32.87	38.76	50.19	42.20	32.92	43.94
TwitterDS-1_FT	41.08	50.14	36.89	**41.86**	50.19	48.88	36.51	**46.46**
TwitterDS-2	54.10	64.69	39.07	**54.01**	65.65	63.14	49.50	**61.16**

We are leaving the discussions of comparing the results of these three Turkish NER models, together with the discussions of the results on different Turkish tweets test sets and their variations for overall comparison to Section 5.1.4, since it will be healthier to compare them all after we report the results of all our Turkish NER models, i.e. with and without the capitalization feature, with and without tweets normalization.

5.1.2. Not Using Capitalization as a Feature

In this subsection, we will present both the phrase-level and token-level performance results of our three different Turkish NER models, this time without using the capitalization feature as a clue for recognizing named entities. Again, the below performance results are the test results obtained on the five different annotated Turkish Twitter data sets. In Table 5.4, Table 5.5 and Table 5.6, we have reported our performance results of NER Model Web, NER Model Tweets and NER Model Web+Tweets respectively, where this time we did not use capitalization as a feature for all three models.

As before, we reported both phrase-level, which are evaluated based on the CoNLL metric, and token-level performance results of the Turkish NER models we explored. Note that all of our NER systems are evaluated only on ENAMEX type named entities, or PLOs. In addition to the overall PLO performances, we are also reporting phrase-level and token-level results for each type of named entities, namely person, location and organization entities throughout this experiments and results chapter.

Table 5.4. Phrase-level and token-level F-score performance results of NER Model Web on Turkish Twitter test sets, without using the capitalization feature.

NER Model Web	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	30.25	48.12	34.20	34.56	43.42	47.62	34.01	41.82
TwitterDS-1	30.25	61.50	37.27	38.52	43.42	59.47	36.77	44.98
TwitterDS-1_LTOC_FT	34.14	48.82	34.98	37.26	48.82	48.22	34.84	44.98
TwitterDS-1_FT	34.14	62.28	38.22	**41.63**	48.82	60.13	37.77	**48.51**
TwitterDS-2	45.12	72.91	44.77	**54.09**	63.69	70.12	55.49	**63.64**

Table 5.5. Phrase-level and token-level F-score performance results of NER Model Tweets on Turkish Twitter test sets, without using the capitalization feature.

NER Model Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	24.30	45.24	27.49	28.32	33.39	45.41	27.77	33.64
TwitterDS-1	24.30	58.82	29.33	31.57	33.39	57.27	29.48	36.24
TwitterDS-1_LTOC_FT	29.79	45.92	30.00	32.61	41.11	46.00	30.39	39.02
TwitterDS-1_FT	29.79	59.59	32.29	**36.43**	41.11	57.92	32.54	**42.10**
TwitterDS-2	40.12	69.09	33.54	**48.15**	60.90	66.10	48.32	**59.49**

Table 5.6. Phrase-level and token-level F-score performance results of NER Model Web+Tweets on Turkish Twitter test sets, without using the capitalization feature.

NER Model Web+Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC	33.93	46.29	33.60	36.27	46.65	46.04	35.58	43.98
TwitterDS-1	33.93	60.25	37.79	40.18	46.65	58.39	38.55	47.11
TwitterDS-1_LTOC_FT	39.14	47.09	35.48	39.62	52.50	46.73	36.36	47.34
TwitterDS-1_FT	39.14	61.15	38.87	**44.00**	52.50	59.16	39.53	**50.87**
TwitterDS-2	52.06	72.40	37.76	**55.45**	68.25	70.13	51.53	**64.95**

5.1.3. Tweet Normalization

In this subsection, we will present both the phrase-level and token-level performance results of our Turkish NER models by applying text normalization on tweets as a preprocessing step before testing. These test results are obtained on the normalized versions of the five different Turkish Twitter data sets annotated with named entities. Note that we will report the performance results on the normalized Twitter test sets with two different groups of models where we used capitalization as a feature while training the first group of Turkish NER models, and then did not use capitalization as a feature while training the second group of our NER models. Within each such group, we have three different NER models again, based on the source text type of the used word embeddings.

At first, we explored both the phrase-level and token-level performance results of our Turkish NER models that use the capitalization clue as a valid feature to recognize named entities, on the normalized Turkish Twitter data sets. We presented their results in Table 5.7, Table 5.8 and Table 5.9 for NER Model Web, NER Model Tweets and NER Model Web+Tweets, respectively. Since the Turkish Twitter text normalization scheme we applied is dealing with the frequently missing capitalizations issues in tweets, we are expecting that this time turning the capitalization feature off will not help, and even make the performance worse compared to when capitalization is on, as it is the case with formal text types such as news articles where we already have proper initial capitalizations for PLO type named entities.

Table 5.7. Phrase-level and token-level F-score performance results of NER Model Web on normalized Turkish Twitter test sets, using the capitalization feature.

NER Model Web	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	31.56	46.45	42.76	37.80	40.96	48.28	40.00	42.03
TwitterDS-1_Norm	31.56	59.52	46.48	41.82	40.96	59.64	43.24	45.40
TwitterDS-1_LTOC_FT_Norm	37.10	47.09	43.84	41.34	48.23	48.84	41.47	46.38
TwitterDS-1_FT_Norm	37.10	60.24	47.79	**45.74**	48.23	60.25	44.99	**50.09**
TwitterDS-2_Norm	52.40	66.11	45.67	**55.20**	63.85	65.17	54.07	**61.88**

Table 5.8. Phrase-level and token-level F-score performance results of NER Model Tweets on normalized Turkish Twitter test sets, using the capitalization feature.

NER Model Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	32.14	44.44	42.81	37.65	40.73	46.61	40.97	41.84
TwitterDS-1_Norm	32.14	57.68	46.42	41.54	40.73	58.00	44.26	45.06
TwitterDS-1_LTOC_FT_Norm	38.47	45.06	45.13	41.94	49.08	47.14	43.06	46.99
TwitterDS-1_FT_Norm	38.47	58.37	49.20	**46.27**	49.08	58.59	46.75	**50.61**
TwitterDS-2_Norm	49.71	66.12	37.02	**52.23**	62.63	65.67	48.56	**60.21**

Table 5.9. Phrase-level and token-level F-score performance results of NER Model Web+Tweets on normalized Turkish Twitter test sets, using the capitalization feature.

NER Model Web+Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	33.90	44.97	44.05	38.97	43.38	46.01	41.81	43.44
TwitterDS-1_Norm	33.90	57.68	47.94	42.79	43.38	57.14	45.28	46.67
TwitterDS-1_LTOC_FT_Norm	39.44	45.60	45.01	42.44	50.75	46.54	43.30	47.81
TwitterDS-1_FT_Norm	39.44	58.37	49.10	**46.61**	50.75	57.72	47.06	**51.37**
TwitterDS-2_Norm	57.42	65.55	43.35	**56.79**	67.18	63.75	52.53	**62.80**

Secondly, this time we used our Turkish NER models that are trained without using the capitalization feature and explored their performance on the normalized versions of our annotated Turkish tweets sets. We presented these results in Table 5.10, Table 5.11 and Table 5.12 for NER Model Web, NER Model Tweets and NER Model Web+Tweets, respectively. It ıs observed that on normalized tweets sets, these results without capitalization in general are not as good as the results we obtained with the capitalization feature used as a strong clue for named entities. This situation is consistent with our expectations since we know that the capitalization feature helps for NER models tested on formal text types, and our normalization scheme partially converts the informal texts into more structured formal texts. Moreover, the results show that Turkish text normalization on tweets before using the NER system improves the performance results obtained on these normalized tweets sets for almost all Turkish NER models we explored.

Table 5.10. Phrase-level and token-level F-score performance results of NER Model Web on normalized Turkish Twitter test sets, without using the capitalization feature.

NER Model Web	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	29.11	45.71	42.36	36.43	40.27	45.78	40.81	41.43
TwitterDS-1_Norm	29.11	58.77	45.98	40.50	40.27	57.48	44.18	44.81
TwitterDS-1_LTOC_FT_Norm	32.91	46.32	44.69	39.63	45.13	46.29	42.95	44.75
TwitterDS-1_FT_Norm	32.91	59.45	48.78	**44.17**	45.13	58.05	46.74	**48.54**
TwitterDS-2_Norm	47.70	72.26	43.40	**54.75**	64.85	69.57	53.14	**63.53**

Table 5.11. Phrase-level and token-level F-score performance results of NER Model Tweets on normalized Turkish Twitter test sets, without using the capitalization feature.

NER Model Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	27.01	46.56	42.49	35.34	37.25	47.01	41.19	40.03
TwitterDS-1_Norm	27.01	59.72	46.10	39.31	37.25	58.55	44.60	43.33
TwitterDS-1_LTOC_FT_Norm	33.92	47.19	44.97	40.35	45.84	47.53	43.49	45.53
TwitterDS-1_FT_Norm	33.92	60.42	49.09	**44.91**	45.84	59.13	47.36	**49.31**
TwitterDS-2_Norm	41.62	70.00	33.85	**49.43**	60.91	66.78	47.46	**59.52**

Table 5.12. Phrase-level and token-level F-score performance results of NER Model Web+Tweets on normalized Turkish Twitter test sets, without using the capitalization feature.

NER Model Web+Tweets	**Phrase-level**				**Token-level**			
Test Set	**PER**	**LOC**	**ORG**	**Overall**	**PER**	**LOC**	**ORG**	**Overall**
TwitterDS-1_LTOC_Norm	31.10	43.86	43.97	37.15	43.53	44.69	42.86	43.57
TwitterDS-1_Norm	31.10	57.21	47.84	41.04	43.53	56.47	46.50	46.77
TwitterDS-1_LTOC_FT_Norm	36.82	44.56	44.93	40.93	49.51	45.29	43.58	47.07
TwitterDS-1_FT_Norm	36.82	58.01	49.00	**45.27**	49.51	57.14	47.40	**50.68**
TwitterDS-2_Norm	52.91	72.91	37.98	**56.12**	68.53	70.57	50.20	**64.88**

5.1.4. The Effects of Word Embeddings Source, Capitalization, Normalization and Used Test Sets on NER Performance

In this section, we will compare and discuss the results of our various Turkish NER models we presented so far, together with the discussions of the variations in results on different Turkish tweets test sets for overall comparison. Until now, we explored four main model parameters or experimental setup parameters affecting the performance results of our NER system. These are namely the variations in Twitter test sets, the source of corpus used for word embeddings, the capitalization feature and finally the normalization scheme.

Table 5.13. Phrase-level and token-level overall performance results to observe the effects of word embeddings source, capitalization and normalization on different Twitter test sets.

Test Set	**Cap**	**Phrase-level (Overall)**			**Token-level (Overall)**		
		Web	**Twt**	**Web+Twt**	**Web**	**Twt**	**Web+Twt**
TwitterDS-1_LTOC	ON	33.68	**32.51**	35.30	38.12	**36.60**	39.57
	OFF	**34.56**	28.32	**36.27**	**41.82**	33.64	**43.98**
TwitterDS-1_LTOC_Norm	ON	**37.80**	**37.65**	**38.97**	**42.03**	**41.84**	43.44
	OFF	36.43	35.34	37.15	41.43	40.03	**43.57**
TwitterDS-1	ON	36.55	**35.14**	38.11	40.42	**38.69**	41.83
	OFF	**38.52**	31.57	**40.18**	**44.98**	36.24	**47.11**
TwitterDS-1_Norm	ON	**41.82**	**41.54**	**42.79**	**45.40**	**45.06**	46.67
	OFF	40.50	39.31	41.04	44.81	43.33	**46.77**
TwitterDS-1_LTOC_FT	ON	**37.37**	**37.42**	38.76	42.51	**42.20**	43.94
	OFF	37.26	32.61	**39.62**	**44.98**	39.02	**47.34**
TwitterDS-1_LTOC_FT_Norm	ON	**41.34**	**41.94**	**42.44**	**46.38**	**46.99**	**47.81**
	OFF	39.63	40.35	40.93	44.75	45.53	47.07
TwitterDS-1_FT	ON	40.53	**40.44**	41.86	45.06	**44.61**	46.46
	OFF	**41.63**	36.43	**44.00**	**48.51**	42.10	**50.87**
TwitterDS-1_FT_Norm	ON	**45.74**	**46.27**	**46.61**	**50.09**	**50.61**	**51.37**
	OFF	44.17	44.91	45.27	48.54	49.31	50.68
TwitterDS-2	ON	53.14	47.72	54.01	60.23	56.68	61.16
	OFF	**54.09**	**48.15**	**55.45**	**63.64**	**59.49**	**64.95**
TwitterDS-2_Norm	ON	**55.20**	**52.23**	**56.79**	61.88	**60.21**	62.80
	OFF	54.75	49.43	56.12	**63.53**	59.52	**64.88**

In Table 5.13, we combined all of the overall results we obtained so far with all of our Turkish NER systems, by varying each of the four parameters mentioned above in order to observe and compare their effects on NER performance. Note that the bold entries within each column indicate better results between two options of capitalization, either turned on so that it is used as a feature by a NER model, or turned off so that it is not used, on their corresponding test set. Moreover, the underlined bold entries show the best result for the corresponding test set, with varying source text type of word embeddings and also varying capitalization. We have one bold and underlined performance result for phrase-level F-score and one bold and underlined entry for token-level F-score for each test set presented in Table 5.13.

To begin, let us discuss first the variations of performance results reported on different Turkish tweets test data sets. As we explained in Section 3.3.2, we mainly have two different data sets for testing our NER system. These original data sets are namely TwitterDS-1 composed of around 5K tweets with 1366 tagged PLOs and TwitterDS-2 composed of around 2.3K tweets having 980 tagged PLOs. Since we have observed some problems in TwitterDS-1, we manually created modified versions of this tweets data set.

The first modified version of TwitterDS-1 is where we corrected around 50 entities annotated as location but actually referring to football club names by replacing the location tags with organization tags, and called this set TwitterDS-1_LTOC. We did this for the sake of making the test data more realistic that reflects the actual usage of named entities on microblog texts. Since such entities with metonymic readings are harder to be recognized correctly for a NER system especially on tweets, the overall performance results show that all of our NER systems perform better on TwitterDS-1 than on TwitterDS-1_LTOC as expected. To be more precise, the phrase-level performance results on TwitterDS-1 and on TwitterDS-1_LTOC differ around 3-4%.

The second modification we performed on TwitterDS-1 is that, we manually removed all completely non-Turkish tweets that include no Turkish words, not even one, in order to have a healthier and fully Turkish Twitter data for our Turkish NER system. We performed this modification on both TwitterDS-1 and TwitterDS-1_LTOC, and called them TwitterDS-1_FT and TwitterDS-1_LTOC_FT respectively. As seen from the results, making the test set composed of only Turkish tweets increases the phrase-level NER performance around 3-5% for both TwitterDS-1 and TwitterDS-1_LTOC, which is also expected.

Although we performed those modifications on TwitterDS-1 in order to increase the representativeness of the test set, we still have problems with this data set since certain portions of named entities are missed to be annotated, which causes a decrease in performance for the NER system. We can see its effects when we compare the results on the two original data sets, TwitterDS-1 and TwitterDS-2. We observed that around 14-18% phrase-level performance improvements are obtained on TwitterDS-2 compared to the results on TwitterDS-1. Even if we compare TwitterDS-2 with the modified version of TwitterDS-1 that performs best, which is the fully Turkish version called TwitterDS-1_FT, we still have around 11-13% phrase-level performance improvements obtained on TwitterDS-2 compared to TwitterDS-1_FT. We have seen that TwitterDS-2 is a better Turkish Twitter test set with properly tagged named entities, and hence we should consider the results on this set in order to better assess the performance of our NER system.

Another important point to discuss here is the effects of word embeddings source corpus on the performance of our Turkish NER system. In Table 5.13, the columns with name Web, Twt and Web+Twt are reserved for the results of our basic three NER models called NER Model Web, NER Model Tweets and NER Model Web+Tweets, respectively. The only difference between them is the text type of large Turkish corpus used for training word embeddings at the unsupervised stage which are then used as a crucial feature at the supervised stage of our NER system. In NER Model Web, we used BOUN Web Corpus in Turkish whereas in NER Model Tweets, we used our own combined Tweets Corpus in Turkish to attain Turkish word embeddings. In NER Model Web+Tweets, we combined these two corpora from Turkish web news articles and tweets, and used it as a source corpus for word embeddings.

We have seen from the results of these three models that, on TwitterDS-1 and on all its variations, the NER Model Web+Tweets generally performed slightly better than the other two models, and the NER Model Tweets generally performed worse than the other two models. However, note that the differences on phrase-level performances between these three models are around 1-2% in general. We cannot directly conclude that adding Turkish tweets corpus to the Turkish news corpus as an additional source for word embeddings significantly improves the performance of the final NER system, but it helps for sure.

If we compare the performance results of these three models on TwitterDS-2, we observe that there is a significant drop in phrase-level performances, around 3-6%, when we only use the Tweets Corpus, compared to the other two models. Although the results of NER Model Web and NER Model Web+Tweets are very close with around 1% difference of phrase-level performance, our best model on TwitterDS-2 is again the NER Model Web+Tweets.

We may explain the performance drop of NER Model Tweets compared to NER Model Web with the data size differences of these two corpora since the number of words in our Tweets Corpus is nearly 60% of the number of words in the BOUN Web Corpus, which may affect the representativeness of word embeddings. However, this size difference may not explain achieving close results with NER Model Web and NER Model Web+Tweets, where NER Model Web+Tweets performed slightly better both on TwitterDS-2 and on TwitterDS-1 and its variations in general.

The reason behind such results can be the fact that although we are changing the source of Turkish word embeddings at the unsupervised stage, at the supervised stage we are still training all of these NER models on an annotated Turkish news data set instead of tweets data set due to the lack of large amount of annotated Turkish tweets data for training. Since those Turkish word embeddings are used as a feature during this training at the supervised stage, the model parameters are tuned based on the news data at the training, and not based on tweets data. Therefore, adding a tweets corpus in addition to the news corpus for word embeddings may not be as effective as we desired while testing on tweets data. We believe that in the future if we have a large amount of annotated Turkish tweets, than with high reliability we can easily train these NER models on huge Turkish Twitter data at the supervised stage and then can observe better the effects of adding Turkish tweets corpus to the Turkish news corpus as a source for word embeddings.

Using the capitalization in texts as a clue for named entities is a valuable feature for the NER task, especially for formal text types. In our general NER model framework, we explained the capitalization feature we used which basically takes into account the capitalization pattern of tokens in the current window. In formal text types such as news articles, proper capitalization of named entities is very common. However, proper capitalizations of named entities are often missed in informal text types such as tweets. With this motivation, for all of our Turkish NER models, we created two version of each model where we keep the capitalization feature in the first case and removed it during the training in the second case. We have reported the performance results of both versions for each Turkish NER model in Table 5.13 with the Cap column, where ON means it is used as a feature during training and OFF means it is not used, in order to compare the effects of using capitalization.

Since the results have shown that NER Model Web+Tweets performed better on all variations of TwitterDS-1 without normalization, let us discuss the effects of capitalization on this test set based on this model. We observed that when we turn the capitalization feature off during the training at the supervised stage, the phrase-level performance improves around 1-2% on all four test sets as variations of TwitterDS-1 without normalization, compared to the cases with the capitalization feature is on. Again, since we observed that NER Model Web+Tweets performed better on TwitterDS-2 without normalization, let us discuss the effects of capitalization on this test set based on this model. We observed that again the phrase-level performance increases around 1% on not normalized TwitterDS-2 when the capitalization feature is not used compared to the case where we use it during training. These results are consistent with our expectations since, especially with word embeddings from lowercased corpus, using the capitalization pattern as a clue for named entities during the training can be misleading when we test on tweets data, where we do not always have named entities with proper capitalization.

Note that the effect of the capitalization feature is completely different when we have normalized versions of Twitter data as test sets for our NER models. We know that on formal text types such as news articles where we already have proper capitalizations for ENAMEX type named entities, using the capitalization as a clue for named entities improves the performance of a NER system. Since the Turkish Twitter text normalization scheme we applied includes a letter case transformation stage, frequently missing capitalizations issues in tweets will be resolved to some extend after normalization. Therefore, we were expecting that on normalized Turkish tweets data, turning the capitalization feature off may not help, and may even decrease the performance compared to models where the capitalization feature is used. In Table 5.13, it is observed that for all of our three NER models and on all of our five normalized Twitter test sets, the phrase-level performance results obtained by using the capitalization feature are always better than the phrase-level result obtained without using it. To be more precise, turning the capitalization feature off decreases the phrase-level performance results around 1-3% on normalized Turkish Twitter data. This is consistent with our expectations since our Turkish text normalization scheme partially converts the informal texts into more structured formal texts on which using the capitalization feature is known to be useful for the NER task. All in all, we can conclude that if you have no normalization on tweets data, it is better to turn the capitalization feature off, whereas if you use a text normalization scheme on tweets, it is better to keep the capitalization feature in order to increase the performance of Turkish NER, especially with word embeddings.

The final discussion we will present here is the effects of Turkish text normalization on the final performance of our Turkish NER models on Twitter. It is observed in Table 5.13 that Turkish text normalization scheme applied on tweets as a preprocessing step for the NER system improves the performance results obtained for almost all Turkish NER models we explored with all normalized tweets sets. To be more precise, if we take into account both of our two best models, NER Model Web and NER Model Web+Tweets, we observe that when the capitalization is on, the phrase-level performance results are increased by around 4-6% on the normalized versions of TwitterDS-1 and all its variants, compared to the results on not normalized versions of these test sets. When the capitalization is off, we still have an improvement with normalization but this time the phrase-level performance results of these NER systems increase around 1-3% on the normalized version of all TwitterDS-1 variants. Similarly, on the normalized version of TwitterDS-2, compared to the not normalized version of this set, we have an improvement on phrase-level performance results by around 2-3% when the capitalization is on, and by around 1-2% when the capitalization is off, again with NER Model Web and NER Model Web+Tweets.

We can conclude that for both capitalization on and capitalization off cases, and on all of our Turkish Twitter test sets, tweets normalization improves the performance, but the range of improvement differs such that when we use the capitalization as a feature, we benefit more from the Turkish tweets normalization scheme. This is consistent with our expectations since the normalizer can correct some of the missing capitalizations in Turkish tweets and the effect of this correction on NER performance will be higher when we use the capitalization as a clue for named entities.

Note that the Turkish NER performance improvements with normalization are higher on TwitterDS-1 and its variants than on TwitterDS-2, i.e. around 4-6% and around 2-3%, respectively. We believe that this situation can have two reasons behind. Firstly, since TwitterDS-1 and even its best variant have problems like missed named entities without annotations, whereas TwitterDS-2 is a much healthier Turkish tweets set with proper annotations of named entities, before normalization we have around 16% performance differences on TwitterDS-2 and on TwitterDS-1. Even with the results on the best variant of TwitterDS-1, which is the fully Turkish version called TwitterDS-1_FT; we still have around 13% performance differences compared to the results on TwitterDS-2 before normalization. This means that there is much more space for improvements on TwitterDS-1 than on TwitterDS-2, which can explain the differences in normalization scheme effects on the NER performance for these two test sets. Secondly, since the researchers [43] that prepared the first annotated Turkish Twitter data set, TwitterDS-1, are the same researchers that proposed this normalizer we used, it might be the case that some rules used in this Turkish tweets normalizer are inspired by this tweets data set, and hence this normalizer works better on TwitterDS-1 than on TwitterDS-2.

As a final remark, after all these comparisons and discussions, we have obtained the best performance results of our Turkish NER system by using NER Model Web+Tweets in general, with keeping the capitalization feature on and applying Turkish tweets normalization. The best phrase-level performance of our Turkish NER system is reported as around 57% on the normalized version of TwitterDS-2 and around 47% on the fully Turkish and normalized version of TwitterDS-1.

5.1.5. The Effects of Using Word Embeddings Compared to Normalization

In this section, we will compare and discuss the results of our Turkish NER models we presented so far, which benefits highly from word embeddings, with the results of our baseline Turkish NER model trained without word embeddings. We will compare the performance results on different Turkish tweets test sets, both without normalization and with normalization for overall comparison. We intend to show here the real effects of the word embeddings we employed on the final performance of our NER models. We will compare the improvements achieved by using word embeddings versus by applying text normalization on Turkish tweets.

In Table 5.14, we have reported both the phrase-level and token-level performance results of our NER systems on three different Twitter data sets. These test sets are namely TwitterDS-1 that is the first and original labeled Turkish tweets data presented in [29], TwitterDS-1_FT which is the improved version of TwitterDS-1 filtered by removing fully non-Turkish tweets, and TwitterDS-2 that is the original set presented in [9]. The NER models are built by adding different features or performing normalization at each step on top of the baseline model. Our baseline NER model for comparison is built by adding only language independent and generic features as explained in Section 3.2.2. These are the local features such as context features, word type information, affixes, and previous tags. Note that we did not use the capitalization feature in the baseline NER model in order to assess its effects on Twitter data.

In the first part, i.e. the top four rows of the results presented in Table 5.14, we added only the normalization stage to our baseline NER system, and then added only the word embeddings feature, denoted with *WordE (W&T)* below, where we used Turkish Web and Tweets corpus together to learn the word representations, which has been shown to perform better than using only the Web corpus before, and finally applied both normalization on Twitter data and used the word embeddings feature together, all without the capitalization feature. In the second part, i.e. the last four rows of the results shown in Table 5.14, we followed the same process but this time by using the capitalization feature in addition to our baseline NER model.

Table 5.14. Phrase-level and token-level overall F-score performance results to show the effects of using word embeddings compared to normalization on different Twitter sets.

	Phrase-level (Overall)			Token-level (Overall)		
NER Model	***Twitter DS-1***	***Twitter DS-1_FT***	***Twitter DS-2***	***Twitter DS-1***	***Twitter DS-1_FT***	***Twitter DS-2***
Baseline (BL)	22.16	25.98	35.16	26.06	30.46	41.95
BL + **Norm**	33.05	39.23	37.17	36.26	42.66	44.58
BL + **WordE** (W&T)	40.18	44.00	55.45	47.11	50.87	64.95
BL + **WordE** (W&T) + **Norm**	**41.04**	**45.27**	**56.12**	46.77	50.68	64.88
Baseline (BL) + Cap	27.16	30.21	37.32	30.35	33.86	43.48
BL + Cap + **Norm**	36.70	40.78	42.18	39.03	43.66	49.95
BL + Cap + **WordE** (W&T)	38.11	41.86	54.01	41.83	46.46	61.16
BL + Cap + **WordE** (W&T) + **Norm**	**42.79**	**46.61**	**56.79**	46.67	51.37	62.80

Let us discuss first the models without the capitalization feature for Turkish Twitter NER. As shown in Table 5.14, we gain a lot in terms of F-score performance when we employ only the word embeddings on top of our baseline NER model. To be more precise, we achieved an improvement of phrase-level performance by 18% on TwitterDS-1 and TwitterDS-1_FT, and by 20% on TwitterDS-2 by adding only the Turkish word embeddings feature, where we used the Turkish Web corpus and Turkish Tweets corpus together, on top of our baseline NER system. On the other hand, if we apply only the Turkish text normalization scheme, which is specialized for social media texts, on our Twitter data sets as a pre-step for NER, we achieved phrase-level F-score performance improvements of around 11% on TwitterDS-1, 13% on TwitterDS-1_FT and 2% on TwitterDS-2 compared to the baseline.

Note that without capitalization, adding both the word embeddings and normalization on top of our baseline NER model results in the best phrase-level performance on all three Twitter data sets. However, the effect of normalization on performance is much less now, when we already have the word embeddings in our NER model. For instance, without capitalization, the normalization on top of word embeddings improves the phrase-level performance results by only 1% on all three Turkish tweets sets, where the improvements were around 11-13% on TwitterDS-1 variations when we apply normalization on top of the baseline model without word embeddings. This indicates that our word embeddings feature already covers some of the effects of normalization, since the unstructured words with typos and all other informality problems are already automatically placed close to their normalized forms in vector space.

Let us now discuss the models with the capitalization feature for Turkish Twitter NER. As shown in the last four rows of Table 5.14, we still improve a lot in terms of NER performance when we use only the word embeddings on top of our baseline NER model with capitalization feature. To be more precise, we achieved an improvement of phrase-level performance by around 11% on TwitterDS-1 and TwitterDS-1_FT, and by 17% on TwitterDS-2 by adding only the Turkish word embeddings to our baseline NER system with capitalization. However, if we apply only the Turkish social media text normalization scheme on our Twitter sets before the NER, we achieved phrase-level performance improvements of around 9-10% on TwitterDS-1 and TwitterDS-1_FT, and 5% on TwitterDS-2 compared to the baseline with capitalization.

Once again, adding both the word embeddings and normalization to our baseline NER model with capitalization results in the best phrase-level performance on all Twitter sets, and even slightly better than the same model without capitalization. In contrast, the effect of normalization on performance is lessened now, when we already have the word embeddings. To illustrate, with capitalization, the normalization on top of word embeddings improves the phrase-level performance results by nearly 5% and 3% on TwitterDS-1 and TwitterDS-2 respectively, where the improvements were around 9% and 5% on TwitterDS-1 and TwitterDS-2 respectively, when we apply normalization on top of the baseline model plus capitalization without word embeddings. This shows that our word embeddings feature still covers the certain portion of the effects of normalization, but this portion is less with capitalization compared to the case without capitalization.

Note that all NER models in Table 5.14, except the one with adding only word embeddings on top of the baseline, benefit from the capitalization feature. This is expected when we apply a normalization scheme that has a capitalization correction stage using lexical resources. However, if we compare the effect of the capitalization feature on the baseline and the baseline plus word embeddings models, we observe that if you employ word embeddings then it is better to turn off the capitalization. This makes sense since we already lowercased our large corpus we used to attain word embeddings. Lowercasing the corpus is recommended previously in order to limit the number of words in the vocabulary and hence decreases the sparsity problem together with increased efficiency for our unsupervised stage. Since the lowercasing step is shown to increase the representativeness of the obtained word embeddings, we keep this as it is. However, if you have a normalizer stage in your NER system, the best results are achieved with normalization and capitalization together with the word embeddings, as shown in the phrase-level NER performance results in the last row of Table 5.14.

All in all, in this section we have shown that using word embeddings that are learnt with an unsupervised approach from a large unlabeled corpus composed of web articles and tweets is much better than applying a Twitter specific text normalizer in terms of NER performance. This is a very important conclusion to draw, since building a text normalizer, especially for informal text types such as microblog texts, requires a lot of language dependent and domain specific features and rules, together with extensive lexical resources that are manually constructed for specific languages and domains. Moreover, for morphologically rich languages like Turkish, text normalization for unstructured data is a challenging task and requires successful morphological analysis. On the other hand, extracting word embeddings from a large unlabeled corpus in a certain language is a lot easier and yet more effective than the much more complex text normalization. There is no dependency on language specific analyzers or rules for word embeddings; the only necessity is to have a large corpus, without any manual tagging, in a certain language in order to automatically learn the semantic relations between words and certain morphological patterns in that natural language. Moreover, tailoring your attained word embeddings for a certain text domain is more straightforward with word embeddings, which requires only having a large unlabeled corpus from that specific domain to increase the representativeness of word embeddings. In our case, we have shown in the previous sections that even by adding a comparably less amount of corpus composed of Turkish tweets into our larger corpus composed of Turkish web articles, the observed improvements on Turkish Twitter data are promising.

5.2. Experiments with NER Models Trained on Twitter Data

Note that until now, we presented our results of Turkish NER models that are trained on a large annotated Turkish news data set, and tested on a relatively small annotated Turkish tweets data sets. Although an ideal Turkish NER model designed to work properly on Turkish Twitter data set should be trained also on similar informal text types, we preferred to train on Turkish news data due to the lack of a large annotated Turkish tweets data set for training.

Although we have a limited amount of annotated Turkish Twitter data, our second attempt here is to train a NER model on this relatively small amount of tweets data with 10-fold cross-validation and compare its results with our first attempt. The purpose of performing 10-fold cross-validation here is to prevent the overfitting problem, which is especially likely when the size of the training data set is small. Throughout this section, we will present the performance results of our Turkish NER models trained on two different annotated Turkish Twitter data sets prepared by Küçük *et al.* [9] and Çelikkaya *et al.* [29].

Based on the previous results we presented, here we will only reproduce NER models that performed better before. To be more precise, we will not reproduce NER models using only Turkish Twitter corpus as a source for word embeddings since we have observed that this model performed worse compared to the ones using the Turkish web corpus or the Turkish web plus tweets corpus for obtaining the word embeddings. Moreover, since our results have shown that NER models that did not use capitalization as a feature during training perform better than the ones using capitalization on not normalized Twitter data, we will only reproduce the models without capitalization from the not normalized tweets data sets at this stage. Furthermore, since we have observed that models using capitalization generally result in better performance on normalized tweets, we will only reproduce the models with the capitalization feature turned on for training on the normalized Twitter data. Finally, among the five different annotated Turkish Twitter data sets we explained in Section 3.3.2, we will use the best two of them, based on our previous performance results on them, as training set for the following experiments. These tweets data sets are namely TwitterDS-1_FT that is the fully Turkish version of TwitterDS-

1, the original annotated set presented in [29], where we manually eliminated the non-Turkish tweets, and TwitterDS-2 that is presented in [9].

Since we have limited amount of tweets within the two different annotated data sets, we used one set for training and validation with 10-fold cross-validation, and we used the other set for testing in this section. Namely, we trained our NER models on TwitterDS-1_FT in order to test on TwitterDS-2, and similarly we trained the other NER models on TwitterDS-2 in order to test on TwitterDS-1_FT. Ideally, if the training set and test set come from different partitions of the same data set with large amount of tweets, the performance results will reflect the real nature of the whole set. However, we are restricted with the number of tweets and PLO samples here and we have to somehow use both tweets data sets. Note that we mentioned about the differences of these two sets, TwitterDS-1 and TwitterDS-2, and concluded that TwitterDS-2 is a healthier data set with properly annotated named entities for Turkish. Therefore, we expect that the models trained on TwitterDS-2 will perform better than the models trained on TwitterDS-1_FT.

At the beginning, we will report the results we obtained from NER models trained and validated on TwitterDS-2 and tested on TwitterDS-1_FT, all with 10-fold cross-validation. We will start with our best models where we applied normalization as a preprocessing step and hence we turned the capitalization feature on. We have compared these results with the results of our previous models trained and validated on a much larger amount of annotated Turkish news. Moreover, we also experimented with different NER models by using only the unlabeled Turkish web corpus for learning word embeddings and using the unlabeled Turkish tweets corpus in addition to this news corpus to attain word embeddings. Both phrase-level and token-level performance results of these NER models for overall PLOs on TwitterDS-1_FT and their comparison can be found in Table 5.15. Note that for the models trained on tweets, we applied 10-fold cross-validation and hence we reported mean values of F-scores together with standard deviation values over 10 rounds with different partitions of training an validation sets. We have also reported the mean performance results we obtained on the validation sets together with the test set.

Although we have a relatively small number of PLO samples in the annotated tweets set compared to the annotated news set, the performance result presented in Table

5.15 are very promising. To be more precise, the news set used for training and validation is composed of around 500K words and 54K PLOs, whereas the TwitterDS-2 set we used here for training and validation is composed of around 21K words and only 980 PLOs. Despite this large size difference that is crucial for a better training with supervised learning, when the normalization is applied and capitalization feature is used on all models in Table 5.15, we observed an improvement of nearly 3% on phrase-level F-scores of our NER models on TwitterDS-1_FT when we train on Turkish tweets, namely TwitterDS-2, instead of Turkish news.

Table 5.15. Overall F-score performance results of the Turkish NER models with normalization and capitalization that are trained on TwitterDS-2 with cross-validation and tested on TwitterDS-1_FT, compared to the results of models trained on news.

Word Embeddings	**Trained On**	**Norm**	**Cap**	**Test Set**	**10-fold Cross Val**	**Phrase-level (Overall)**	**Token-level (Overall)**
Web	News	Yes	ON	Twitter DS-1_FT	-	**45.74**	50.09
Web	Twitter DS-2	Yes	ON	*Twitter DS-2 (Val)*	***Mean***	*64.62*	*67.40*
					STD	*7.49*	*7.98*
				Twitter DS-1_FT	**Mean**	**48.75**	52.08
					STD	1.13	1.09
Web + Tweets	News	Yes	ON	Twitter DS-1_FT	-	**46.61**	51.37
Web + Tweets	Twitter DS-2	Yes	ON	*Twitter DS-2 (Val)*	***Mean***	*65.26*	*67.46*
					STD	*8.15*	*7.84*
				Twitter DS-1_FT	**Mean**	**48.96**	52.39
					STD	1.22	1.23

Note that the mean validation performance results, presented in Table 5.15, of our NER models trained each time on different 90% training partition of TwitterDS-2 is much higher than the test results, since we used the other 10% validation partition again from TwitterDS-2 and hence, they have a similar nature. However, when we tested these models on the whole TwitterDS-1, which is a larger and completely different test set, the performance results drop as expected. Note also that the standard deviation on the validation results is relatively higher than the standard deviation we have on the test results over 10 rounds. This shows the necessity to apply 10-fold cross-validation on such small data sets so that good or bad results are not obtained just by coincidence on a small test set.

Note also that adding tweets corpus in addition to the news corpus as an unlabeled Turkish text source for learning word embeddings improves the performance results on validation and test sets only very slightly. We believe that if we have a large amount of labeled tweets set for training together with a huge amount of unlabeled tweets corpus with a more representative word representations, we may observe the real effects of the source text type of word embeddings in the future.

Table 5.16. Overall F-score performance results of the Turkish NER models without normalization and without the capitalization feature, which are trained on TwitterDS-2 and tested on TwitterDS-1_FT, and compared with the results of the models trained on news.

Word Embeddings	**Trained On**	**Norm**	**Cap**	**Test Set**	**10-fold Cross Val**	**Phrase-level (Overall)**	**Token-level (Overall)**
Web	News	No	OFF	Twitter DS-1_FT	-	**41.63**	48.51
Web	Twitter DS-2	No	OFF	*Twitter DS-2 (Val)*	***Mean***	*61.92*	*63.99*
					STD	*6.61*	*6.56*
				Twitter DS-1_FT	**Mean**	**41.06**	43.42
					STD	2.32	2.35
Web + Tweets	News	No	OFF	Twitter DS-1_FT	-	**44.00**	50.87
Web + Tweets	Twitter DS-2	No	OFF	*Twitter DS-2 (Val)*	***Mean***	*65.39*	*66.35*
					STD	*8.03*	*7.18*
				Twitter DS-1_FT	**Mean**	**43.43**	45.78
					STD	2.07	2.14

Next, we have presented the results of our models this time where no normalization is applied before the NER and hence the capitalization feature is off, again trained and validated on TwitterDS-2 with cross-validation and tested on TwitterDS-1_FT. We have compared these results with the results of our previous models trained and validated on the large annotated Turkish news set, with the same capitalization and normalization settings. Similarly, we also experimented with different NER models by using only unlabeled Turkish web corpus and using an additional unlabeled tweets corpus. Both phrase-level and token-level performance results of these NER models for overall PLOs on TwitterDS-1_FT and their comparison can be found in Table 5.16.

Despite the large size and number of PLO samples differences between the annotated news data having 54K PLOs and tweets data called TwitterDS-2 having only 980 PLOs we used for training and validation, we found the performance results presented in Table 5.16 still very promising. To be more precise, when the normalization is not applied and the capitalization feature is therefore not used on all models in Table 5.16, we still achieve nearly the same phrase-level F-scores with our NER models on TwitterDS-1_FT when we train on a small amount of Turkish tweets instead of a large amount of Turkish news. Moreover, comparing the results in Table 5.15 and Table 5.16, we observe that applying normalization on Turkish tweets yields in better performance improvements when we train our NER models on Twitter data compared to when we train on news data. Lastly, adding Turkish tweets corpus to the web corpus to attain word embeddings also yields in better performance improvements when we do not apply normalization, which indicates that the word embeddings obtained from tweets corpus help to cover some portion of structural problems we observe in informal texts which are partly corrected after normalization and hence the embeddings from tweets will not help as before, which is also expected.

Similarly to what we observed in Table 5.15, again the mean validation performance results presented in Table 5.16 of our NER models trained and validated on TwitterDS-2 is much higher than the test results we obtained on TwitterDS-1_FT, which is expected since these two sets are shown to have some characteristic differences before. Note also that the standard deviation on the validation results is relatively higher than the standard deviation we have on the test results over 10 rounds. This again shows the necessity to apply 10-fold cross-validation on such small data sets in order not to obtain good or bad results by coincidence with the chosen partitioning.

Up until now, we have experimented with the NER models trained and validated on TwitterDS-2 and tested on TwitterDS-1_FT. Now we will continue with the other way around, i.e. with the NER models trained and validated on TwitterDS-1_FT and then tested on TwitterDS-2. Note that since TwitterDS-1_FT is shown to be a problematic data set with many missing annotations for named entities, TwitterDS-2 is actually a healthier data set with proper annotations. Therefore, we do not expect any improvements and we expect a decrease in NER performance when we train on TwitterDS-1_FT and test on TwitterDS-2, compared to the models trained on news data.

Table 5.17. Overall F-score performance results of the Turkish NER models with normalization and capitalization that are trained on TwitterDS-1_FT with cross-validation and tested on TwitterDS-2, compared to the results of models trained on news.

Word Embeddings	**Trained On**	**Norm**	**Cap**	**Test Set**	**10-fold Cross Val**	**Phrase-level (Overall)**	**Token-level (Overall)**
Web	News	Yes	ON	Twitter DS-2	-	**55.20**	61.88
Web	Twitter DS-1_FT	Yes	ON	*TwitterDS-1_FT (Val)*	***Mean***	*57.33*	*59.32*
					STD	*6.13*	*6.42*
				Twitter DS-2	**Mean**	**44.52**	47.60
					STD	1.35	1.56
Web + Tweets	News	Yes	ON	Twitter DS-2	-	**56.79**	62.80
Web + Tweets	Twitter DS-1_FT	Yes	ON	*TwitterDS-1_FT (Val)*	***Mean***	*58.43*	*59.42*
					STD	*6.61*	*6.37*
				Twitter DS-2	**Mean**	**44.87**	48.09
					STD	1.43	1.66

In Table 5.17 and 5.18, we presented the performance results of the NER models trained and validated on TwitterDS-1_FT and tested on TwitterDS-2, and compared these results with the models trained on news data. In the first case, we applied normalization and turned the capitalization feature on whereas in the second case, we did not apply normalization and hence the capitalization feature is off.

Table 5.18. Overall F-score performance results of the Turkish NER models without normalization and without the capitalization feature that are trained on TwitterDS-1_FT and tested on TwitterDS-2, compared to the results of the models trained on news.

Word Embeddings	**Trained On**	**Norm**	**Cap**	**Test Set**	**10-fold Cross Val**	**Phrase-level (Overall)**	**Token-level (Overall)**
Web	News	No	OFF	Twitter DS-2	-	**54.09**	63.64
Web	Twitter DS-1_FT	No	OFF	*TwitterDS-1_FT (Val)*	***Mean***	*54.86*	*55.04*
					STD	*5.78*	*5.96*
				Twitter DS-2	**Mean**	**40.88**	40.93
					STD	1.43	1.47
Web + Tweets	News	No	OFF	Twitter DS-2	-	**55.45**	64.95
Web + Tweets	Twitter DS-1_FT	No	OFF	*TwitterDS-1_FT (Val)*	***Mean***	*58.33*	*57.40*
					STD	*5.92*	*6.03*
				Twitter DS-2	**Mean**	**43.25**	43.29
					STD	1.64	1.85

As we have observed from the results presented in Table 5.17 and Table 5.18, there are significant drops in the performance results on the same test set TwitterDS-2 when we train our NER models on TwitterDS-1_FT instead of the news data. This performance differences cannot only be explained by the different data size and number of PLO samples in these two different training sets, since we have seen that although TwitterDS-2 is also a small set, we obtained either nearly the same or better performance results when we train our models on it instead of news. This indicates that TwitterDS-1_FT is a problematic annotated tweets data set, which is not suitable for training a NER system. Although this training set is problematic, the mean performance results on the validation sets are much higher since both the training and validation partitions are sampled from the same tweets data, TwitterDS-1_FT, and hence share the same nature. However, this is not enough since we need our NER system to work properly on completely unknown Turkish tweets in the future. By comparing the differences in the performance results with the model trained on TwitterDS-1_FT and TwitterDS-2, we can conclude that having a healthier annotated Turkish tweets data set is crucial for training a NER system in order to succeed on unknown future tweets, together with the necessity of having a sufficient number of PLO samples within this labeled Twitter set.

The experiments we employed in this section, especially the ones where we used a healthier tweets data set, TwitterDS-2, as a training set instead of news data, tells us that if both the training set and test set share a similar nature in terms of text types, such as formal versus informal texts or long versus short texts, we can obtain better NER prediction results as expected. We believe that if we can increase the size of the training data composed of annotated Turkish tweets; the performance will get even better. Therefore it is a must to firstly construct a large amount of Turkish Twitter data annotated with named entities in order to have successful Turkish Twitter NER systems with acceptable prediction accuracies in the future. It is also important to keep this labeled Turkish tweets data up to date for training since the discussed topics and hence the mentioned named entities may vary in time especially for dynamic microblogging environments such as Twitter.

5.3. Comparison with the State-of-the-art

To the best of our knowledge, there have been three studies on Twitter NER for Turkish, which are published very recently by Çelikkaya *et al.* [29] in 2013, by Küçük *et al.* [9] in 2014, and by Küçük and Steinberger [10] in 2014. In this section, we will compare the results of these previous Turkish Twitter NER systems with our proposed model and show that our Twitter NER system for Turkish outperforms the state-of-the-art performance for the same NLP task.

In Table 5.19, we have listed the phrase-level performance results for overall PLOs, i.e. ENAMEX type named entities, on the same Turkish tweets test set called TwitterDS-1, in order to compare the previously proposed Turkish Twitter NER systems with our own system. We have already explained the details of these three recent previous Twitter NER systems for Turkish in Section 2.3. For the sake of a healthier comparison, we tried to report the results with the most similar settings possible for different NER systems. We used the same training set with the first system in our study, but the second NER system uses a different multilingual news data and the third system, which is rule based, does not have a training phase at all. We should compare the results with capitalization on and off, with normalization or with no normalization etc. separately in order to see the real improvements achieved with our Turkish Twitter NER system. Although we keep these parameters and test sets exactly the same, since all previous NER systems use gazetteer lists for named entities whereas our system does not, and since they did not report their results without those gazetteer entities, this model parameter is not exactly the same for all systems in Table 5.19.

Table 5.19. Comparing the phrase-level F-score performance results of previous Turkish Twitter NER systems with our proposed NER system on the same tweets data called TwitterDS-1.

System	**Trained On**	**Test Set**	**Gazetteer**	**Phrase-level (Overall)**			
				No Normalization		**Normalization**	
				Cap ON	**Cap OFF**	**Cap ON**	**Cap OFF**
Çelikkaya *et al.*, (2013)	Turkish News [11]	TwitterDS-1	Yes	12.23	13.88	**19.28**	15.27
Küçük *et al.*, (2014)	EMM News [10]	TwitterDS-1	Yes	29.64	-	-	-
Küçük and Steinberger, (2014)	No Training Phase	TwitterDS-1	Yes	30.11	**36.63**	**-**	**-**
Our NER System	Turkish News [11]	TwitterDS-1	No	38.11	**40.18**	**42.79**	41.04

We have observed from this chart that, even without gazetteers, our proposed Turkish Twitter NER system outperforms the reported results of all previous systems using gazetteers, with the same combination of capitalization and normalization choices, and on the same test set in Turkish called TwitterDS-1. To be more precise, the closest performance results to our system are achieved with the latest study on Turkish Twitter NER by Küçük and Steinberger [10], in which they achieved phrase-level F-score performance of 30.11% and 36.63% when capitalization is on and off respectively, without normalization on TwitterDS-1 and with additional gazetteer features only. We improved those results by around 8% and 4% with the capitalization feature on and off respectively, and again with no normalization, where we achieved phrase-level F-score performance of 38.11% and 40.18% for each setting respectively. Since the only reported result with normalization on TwitterDS-1 is obtained with the initial Turkish Twitter NER system proposed by Çelikkaya *et al.* [29], we significantly outperformed its best phrase-level F_1-score performance of 19.28% when both capitalization and normalization are used, by means of around 24% increase where we achieved 42.79% performance on TwitterDS-1.

Table 5.20. Comparing the phrase-level F-score performance results of previous Turkish Twitter NER systems with our proposed NER system on the same tweets data called TwitterDS-2.

System	**Trained On**	**Test Set**	**Gazetteer**	**Phrase-level (Overall)**			
				No Normalization		**Normalization**	
				Cap ON	**Cap OFF**	**Cap ON**	**Cap OFF**
Küçük *et al.*, (2014)	EMM News [10]	TwitterDS-2	Yes	37.24	-	-	-
			+Extended	42.68	-	-	-
Küçük and Steinberger, (2014)	No Training Phase	TwitterDS-2	Yes	46.55	**47.41**	-	47.92
Our NER System	Turkish News [11]	TwitterDS-2	No	54.01	**55.45**	**56.79**	56.12

In Table 5.20, we performed the similar comparison this time on our second main test set called TwitterDS-2, on which all NER systems obtain better results since we believe it is a healthier set, with properly annotated named entities, than the first set. We have reported the phrase-level performance results for overall ENAMEX type named entities on the same tweets test set, for the sake of NER system comparisons on it.

The results in Table 5.20 show that, even without gazetteers, our proposed Turkish Twitter NER system outperforms the reported results of both previous systems using gazetteers, with the same settings of capitalization and normalization, and on the same TwitterDS-2 test set. To be more precise, the closest performance results compared to our system is achieved again with the latest study by Küçük and Steinberger [10], in which they achieved phrase-level F-score performance of 46.55% and 47.41% when capitalization feature is used and not used respectively, without normalization on TwitterDS-2 and with additional gazetteer features only. We improved both of those results by around 8% with the capitalization feature on and off, and again with no normalization, where we achieved phrase-level F_1-score performance of 54.01% and 55.45% for each setting, respectively. Note that the only reported result with normalization on TwitterDS-2 is achieved with the same latest study [10], where we significantly outperformed its phrase-level F_1-score performance of 47.92% when capitalization is turned off. The improvement we obtained is around 8% increase in the phrase-level F-score performance where we achieved 56.12% performance on TwitterDS-2, with normalization and without capitalization, with our Turkish Twitter NER system. Since no previous results are reported with both capitalization on and normalization, we cannot compare our best result of 56.79% having this setting with any of the previous work.

Finally, without restricting ourselves to keep as similar set of model parameter settings as possible for a healthier comparison between our NER system and the previous systems, we have shown the best performance results within each NER system together with the details of which settings they used to achieve their own best results in Table 5.21. These best performance results are presented for both on TwitterDS-1 and TwitterDS-2 test sets in Turkish. From this chart, we can conclude that the state-of-the-art performance results for Turkish Twitter NER task was achieved by the latest system presented by Küçük and Steinberger [10] where they reported 38.01% F_1-score on TwitterDS-1 and 48.13% F_1-score on TwitterDS-2 with their best model settings. These settings are namely using gazetteers list, with capitalization feature turned off, and with no normalization, together by expanding their gazetteer lists of named entities with diacritics variations.

Table 5.21. Comparing the best phrase-level F-score performance results of previous Turkish Twitter NER systems with our proposed NER system, all with its own best model parameter settings, on two tweets data sets called TwitterDS-1 and TwitterDS-2.

System	Trained On	Best Settings				Test Set	Phrase-level (Overall)
		Gaz.	**Norm.**	**Cap.**	**Other**		
Çelikkaya *et al.*, (2013)	Turkish News [11]	Yes	Yes	ON	-	TwitterDS-1	19.28
Küçük *et al.*, (2014)	EMM News [10]	Yes	No	ON	relaxed & extended gazetteer	TwitterDS-1	36.11
						TwitterDS-2	42.68
Küçük and Steinberger, (2014)	No Training Phase	Yes	No	OFF	diacritics expanded gazetteer	**TwitterDS-1**	**38.01**
						TwitterDS-2	**48.13**
Our NER Systems	Turkish News [11]	No	Yes	ON	word embeddings + filter non-Turkish	TwitterDS-1	46.61
					word embeddings	**TwitterDS-2**	**56.79**
	Turkish Tweets	No	Yes	ON	word embeddings + filter non-Turkish	**TwitterDS-1**	**48.96**
					word embeddings	TwitterDS-2	44.87

Note that our Turkish Twitter NER system outperforms these state-of-the-art results on both Turkish Twitter test sets, even without using gazetteers. We achieved our best performance results when we apply normalization on tweets and keep the capitalization as a feature, with Turkish word embeddings obtained from our Web+Tweets corpus, and with applying the filter on fully non-Turkish tweets on TwitterDS-1. By using Turkish news data as a training set, we achieved 46.61% phrase-level F_1-score on

TwitterDS-1, which is an improvement on the state-of-the-art performance by nearly 9%, and we achieved 56.79% F_1-score for overall ENAMEX type named entities on TwitterDS-2, which outperforms the state-of-the-art result by nearly 9%. Since we actually obtained better performance results on TwitterDS-1 when we train our NER system on limited amount of annotated Turkish Twitter data, TwitterDS-2, with 10-fold cross-validation, we also reported those results in the last columns of Table 5.21. These last two results are actually the mean values of the phrase-level F_1-scores with this 10-fold cross-validation. Even on small amount of training data composed of annotated Turkish tweets, the results are very promising especially on TwitterDS-1 such that we achieved mean phrase-level F-score of 48.96% on TwitterDS-1, which outperforms the state-of-the-art results by nearly 11% on TwitterDS-1. On TwitterDS-2, training on Twitter data did not improve the performance results since this time we used TwitterDS-1 as a training set, which is shown to be problematic. Therefore, our very best results on TwitterDS-2 is still the one we obtained by training a NER model on news data, with a phrase-level F-score of 56.79% as we stated before.

6. CONCLUSION

In the context of this thesis, we investigated the Named Entity Recognition (NER) problem on microblog texts in morphologically rich languages. The target microblogging environment we chose in this study is Twitter since it is the most popular microblogging site where extracting information from has gained high attention recently due to the value of the information it contains. The target language in the scope of this work is Turkish, which is known as a morphologically rich language that creates additional challenges for Natural Language Processing (NLP) tasks such as NER. The scope of this thesis is motivated by the fact that, although it is challenging, there is still a lot of room for improvements for the NLP tasks on Turkish social media data.

In order to accomplish the NER task on a challenging domain composed of informal and unstructured texts in a highly inflectional language, we adopted a semi-supervised learning approach based on neural networks that benefits from continuous representations of words. At the first stage, we attained distributed representations of words in continuous vector space, which are known as word embeddings, by employing a fast unsupervised learning method on a large unlabeled corpus. In the second stage, we exploited these word embeddings together with language independent features in order to train our neural network on labeled data using the averaged multiclass perceptron algorithm. At the end, we evaluated our NER system on Turkish Twitter messages called tweets that are annotated with named entities. We have compared our results on two different Turkish Twitter data sets with the state-of-the-art NER system proposed for Twitter data in Turkish. We have shown that our Turkish Twitter NER system outperforms the state-of-the-art performance results on both data sets.

Our proposed NER system is tailored to perform better on microblogging texts in Turkish such that, at the unsupervised stage where we learn our word embeddings, in addition to using huge amount of unlabeled data composed of Turkish news articles, we also explored using large amount of text corpus composed of Twitter messages in Turkish in order to attain Turkish word embeddings. Since these word embeddings are used as a crucial feature at the supervised stage, we propagated the specific word usages in Twitter messages indirectly to our supervised stage and trained our NER system with that additional information specific to microblogging texts. Moreover, we applied what we called Twitter processing on our tweets data sets in order to tag Twitter and web specific usages such as mentions, hashtags, smileys, URLs etc. Embedding those Twitter specific keywords into our large Turkish Twitter corpus also makes our system specialized towards unstructured Twitter texts. Furthermore, we investigated the benefit of the capitalization feature that is widely used for successful NER systems on formal text types where proper capitalization of named entities is almost always observed and hence the capitalization pattern of words is a useful clue to recognize named entities. We have shown that in order to have a NER system using lowercased word embeddings that performs better on informal text types where we usually lack proper capitalizations; you should turn the capitalization feature off. In addition, we applied a recently proposed Turkish text normalization scheme, which has specifically been designed for social media data, on our tweets data sets and we have shown that it also improves the performance of our NER system on Turkish tweets. This text normalization also involves tagging certain Twitter and web specific keywords such as hashtags, mentions, retweets, smileys, URLs etc. which makes our final system specialized towards microblog texts together with the similar processing performed on unlabeled tweets at the unsupervised stage. Since this normalization scheme also corrects the lacking capitalizations in tweets to some extend, we have shown that if you have such a normalization stage before applying NER on your data, you should turn the capitalization feature on again in order to have even better NER performance. The last adaptation we examined is training our NER system on annotated Turkish tweets. Although most of the previous NER systems on Turkish tweets used large labeled Turkish news data for training due to the lack of enough Turkish tweets data annotated with named entities, we have shown that even if you train your system on a limited number of annotated tweets with 10-fold cross-validation, the results are promising. This indicates the necessity of large amount of annotated Turkish tweets in order to have a successful Turkish Twitter NER system with acceptable performance.

Another important aspect of this study is that, since the only language dependent part of our NER system is the normalization scheme we applied on Turkish tweets, and since even without this normalization we outperform the previous state-of-the-art study on Turkish Twitter NER, we believe that our approach can be easily adapted to other morphologically rich languages. We have shown that utilizing the word representations in a semi-supervised learning approach is highly effective for Twitter NER and can result in state-of-the-art performance even without using any language dependent features and gazetteer lists. If you have a huge amount of unlabeled corpus composed of formal, or even better informal, text types in any language, then you can follow this approach and may improve the performance of Twitter NER especially for morphologically rich languages where there is still room for improvements. Applying social media text normalization for any language on tweets data is optional, but has been shown to be beneficial for further improvements on the final performance of a NER system on microblog texts.

The final and maybe the most important conclusion we draw from this study is that, using word embeddings that are attained by unsupervised learning from a large unlabeled corpus composed of web articles and tweets is much better for the NER performance than applying a Twitter specific text normalization scheme. Building a text normalizer, especially for informal text types, requires a lot of language dependent and domain specific features and rules, together with extensive lexical resources that are manually constructed for a specific language and a specific domain. In addition, for morphologically rich languages such as Turkish, text normalization for unstructured data is even more challenging. In contrast, learning word embeddings from a large unlabeled corpus in certain language is a lot easier and yet more effective than the more complex text normalization. There is no dependency on language specific analyzers or rules for word embeddings. The only requirement is having a large corpus in certain language in order to automatically obtain the semantic relations between words in that natural language. We believe that our approach can be adapted to other morphologically rich languages, especially when successful social media specific text normalizers are not present, in order to improve the NER performance on such unstructured texts. Even if such normalizers are present, we have shown that using word embeddings together with text normalization yields increased performance results for NER systems on microblog texts.

6.1. Future Work

Although we outperformed the state-of-the-art results for the Turkish Twitter NER task, we believe there is still room for improvements. As a future work, we need to construct a huge amount of unlabeled tweets corpus in addition to the news corpus since the size of the corpus we have for learning word embeddings is not comparable to those used in the literature from news domain, which is shown to affect the representativeness of the attained word embeddings and hence the performance of the final NER task. Moreover, we need to construct a large amount of annotated Turkish tweets data in order to train a successful NER system for Turkish tweets and once we have such a training set, we can explore using the Twitter specific features directly at the supervised stage. Although we are restricted with the language independent features in this study, a next step can be investigating language dependent features and gazetteer lists in order to see if they further improve the performance of NER on Turkish tweets.

Another future work can be using this NER system on Turkish tweets as a preprocessing step for Turkish sentiment analysis on Twitter, and investigating the benefits of our NER system as a subtask for sentiment analysis in Turkish. The sentiment analysis task has an increasing popularity and achieves high attention both from industry and academia and since the sentiment of a tweet may often be referring to organizations, brands, companies and person names such as political figures, we believe that our proposed Turkish Twitter NER system can be useful for sentiment analysis on Turkish tweets.

REFERENCES

1. Grishman, R. and B. Sundheim, "Named Entity Task Definition", *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pp. 317–332 (Appendix C), Association for Computational Linguistics, Stroudsburg, PA, USA, 1995.

2. Grishman, R. and B. Sundheim, "Design of the MUC-6 Evaluation", *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pp. 1–11, Association for Computational Linguistics, Stroudsburg, PA, USA, 1995.

3. Chinchor, N. A., "Named Entity Task Definition", *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, p. Appendix E, Fairfax, VA, 1998.

4. Tjong Kim Sang, E. F. and F. De Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition", *Proceedings of the Seventh Conference on Natural Language Learning at Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*, CONLL '03, pp. 142–147, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003.

5. Nadeau, D. and S. Sekine, "A Survey of Named Entity Recognition and Classification", *Lingvisticae Investigationes*, Vol. 30, No. 1, pp. 3–26, 2007.

6. Şeker, G. A. and G. Eryiğit, "Initial Explorations on using CRFs for Turkish Named Entity Recognition", *In Proceedings of the 24th International Conference on Computational Linguistics*, pp. 2459–2474, Mumbai, India, 2012.

7. Hakkani-Tür, D. Z., *Statistical Language Modelling for Turkish*, Ph.D. Thesis, Bilkent University, 2000.

8. Kireyev, K., L. Palen and K. M. Anderson, “Applications of Topics Models to Analysis of Disaster-Related Twitter Data”, *Neural Information Processing Systems (NIPS) Workshop on Applications for Topic Models: Text and Beyond*, 2009.

9. Küçük, D., G. Jacquet and R. Steinberger, “Named Entity Recognition on Turkish Tweets”, *Proceedings of the Language Resources and Evaluation Conference*, 2014.

10. Küçük, D. and R. Steinberger, “Experiments to Improve Named Entity Recognition on Turkish Tweets”, *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL) Workshop on Language Analysis for Social Media*, Gothenburg, Sweden, 2014.

11. Tür, G., D. Hakkani-tür and K. Oflazer, “A Statistical Information Extraction System for Turkish”, *Natural Language Engineering*, Vol. 9, No. 2, pp. 181–210, 2003.

12. Küçük, D. and A. Yazıcı, “A Hybrid Named Entity Recognizer for Turkish”, *Expert Systems with Applications*, Vol. 39, No. 3, pp. 2733–2742, 2012.

13. Küçük, D. and A. Yazıcı, “Named Entity Recognition Experiments on Turkish Texts”, *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, Roskilde, Denmark, 2009.

14. Tatar, S. and I. Çiçekli, “Automatic Rule Learning Exploiting Morphological Features for Named Entity Recognition in Turkish”, *Journal of Information Science*, Vol. 37, No. 2, pp. 137–151, 2011.

15. Yeniterzi, R., “Exploiting Morphology in Turkish Named Entity Recognition System”, *Proceedings of the ACL 2011 Student Session*, Human Language Technologies - Student Session ’11, pp. 105–110, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011.

16. Demir, H., *Semi-Supervised Learning Based Named Entity Recognition For Morphologically Rich Languages*, M.S. Thesis, Boğaziçi University, 2014.

17. Demir, H. and A. Özgür, "Improving Named Entity Recognition for Morphologically Rich Languages using Word Embeddings", *Proceedings of the 13th International Conference on Machine Learning and Applications*, ICMLA '14, pp. 177-122, Detroit, Michigan, USA, 2014.

18. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", *Computing Research Repository*, Vol. 1301, No. 3781, pp. 1–12, 2013.

19. Ritter, A., S. Clark, M. and O. Etzioni, "Named Entity Recognition in Tweets: An Experimental Study", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1524–1534, 2011.

20. Ramage, D., D. Hall, R. Nallapati and C. D. Manning. "Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-labeled Corpora", *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* Vol. 1, pp. 248–256, Morristown, NJ, USA, 2009.

21. Gimpel, K., N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan and N. A. Smith, "Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments", *Association for Computational Linguistics*, 2011.

22. Brown, P. F., P. V. deSouza, R. L. Mercer, V. J. D. Pietra and J. C. Lai. "Class-based n-gram Models of Natural Language", *Computational Linguistics,* 1992.

23. Liu, X., S. Zhang, F. Wei and M. Zhou, "Recognizing Named Entities in Tweets", *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, pp. 359–367, 2011.

24. Liu, X., M. Zhou, F. Wei, Z. Fu and X. Zhou, "Joint Inference of Named Entity Recognition and Normalization for Tweets", *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers,* Vol. 1, pp. 526–535, 2012.

25. Li, C., J. Weng, Q. He, Y. Yao, A. Datta, A. Sun and B. Lee, "TwiNER: Named Entity Recognition in Targeted Twitter Stream", *Proceedings of the 35th International Association for Computing Machinery (ACM) Special Interest Group On Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval*, pp. 721– 730, 2012.

26. Oliveira, D. M., A. H. F. Laender, A. Veloso and A. S. da Silva, "FS-NER: A Lightweight Filter-Stream Approach to Named Entity Recognition on Twitter Data", *Proceedings of the 22nd International Conference on World Wide Web Companion*, pp. 597–604, 2013.

27. Bontcheva, K., L. Derczynski, A. Funk, M. Greenwood, D. Maynard and N. Aswani, "TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text", *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, 2013.

28. Toutanova, K., D. Klein, C. Manning and Y. Singer, "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network", *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT-NAACL 2003, pp. 252-259, 2003.

29. Çelikkaya, G., 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,* 2013.

30. Pouliquen, B. and R. Steinberger, “Automatic Construction of Multilingual Name Dictionaries”, In C. Goutte *et al.*, editor, *Learning Machine Translation*, Advances in Neural Information Processing Systems Series, pp. 59–78, MIT Press, 2009.

31. Zemberek, *Turkish Unique Word List of Zemberek NLP Library for Turkic Languages*, 2010, http://zemberek.googlecode.com/files/full.txt.tr.tar.gz, [Accessed April 2014].

32. Hinton, G. E., J. L. McClelland and D. E. Rumelhart, “Distributed Representations”, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, pp. 77–109, MIT Press, 1986.

33. Rumelhart, D. E., G. E. Hinton and R. J. Williams, “Learning Representations by Back-propagating Errors”, *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

34. Bengio, Y., R. Ducharme, P. Vincent and C. Janvin, “A Neural Probabilistic Language Model”, *Journal of Machine Learning Research*, Vol. 3, pp. 1137–1155, 2003.

35. Collobert, R. and J. Weston, “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, *Proceedings of the 25th International Conference on Machine Learning*, ICML ’08, pp. 160–167, Association for Computing Machinery, 2008.

36. Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, “Natural Language Processing (Almost) from Scratch”, *Journal of Machine Learning Research*, Vol. 12, pp. 2493–2537, 2011.

37. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality”, *Neural Information Processing Systems (NIPS)*, pp. 3111–3119, 2013.

38. Morin, F. and Y. Bengio, “Hierarchical Probabilistic Neural Network Language Model”, *Artificial Intelligence and Statistics (AISTATS)*, pp. 246–252, 2005.

39. Ratinov, L. and D. Roth, “Design Challenges and Misconceptions in Named Entity Recognition”, *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL ’09, pp. 147-155, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009.

40. Freund, Y. and R. E. Schapire, “Large Margin Classification Using the Perceptron Algorithm”, *Machine Learning*, Vol. 37, No. 3, pp. 277–296, 1999.

41. Rosenblatt, F., “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, *Psychological Review*, Vol. 65, No. 6, pp. 386–408, 1958.

42. Kakade, S. M., S. Shalev-Shwartz and A. Tewari, “Efficient Bandit Algorithms for Online Multiclass Prediction”, *Proceedings of the 25th International Conference on Machine Learning*, ICML ’08, pp. 440–447, Association for Computing Machinery, New York, NY, USA, 2008.

43. Torunoğlu, D. and G. Eryiğit, “A Cascaded Approach for Social Media Text Normalization of Turkish”, *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM) at European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, 2014.

44. Sak, H., T. Güngör and M. Saraçlar, "Turkish Language Resources: Morphological Parser, Morphological Disambiguator and Web Corpus", *6th International Conference on Natural Language Processing*, GoTAL 2008, Vol. 5221, pp. 417-427, Springer, 2008.

45. Sezer, B. and T. Sezer, “TS Corpus: Herkes İçin Türkçe Derlem”, *Proceedings of the 27th National Linguistics Conference*, pp. 217-225, Antalya, Turkey, 2013.