# SEMI-SUPERVISED LEARNING BASED NAMED ENTITY RECOGNITION FOR MORPHOLOGICALLY RICH LANGUAGES

by

Hakan Demir

B.S., Computer Engineering, Boğaziçi University, 2013

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

2014

# ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Assist. Prof. Arzucan Özgür, who has supported me throughout my thesis with her patience and knowledge. This thesis would not have been written without her guidance and effort.

My deepest gratitude goes to my family for their unflagging love and support throughout my life. I cannot put into words how much their support means to me and I am beyond fortunate to have them.

I am thankful to all members of TÜBİTAK Speech and Natural Language Processing Lab especially Ahmet Afşin Akın and Yusuf Ziya Işık for sharing their valuable experiences.

Last but not least, I would like to thank my friends for the pleasant time I had with them. I am grateful to have such nice friends. Obviously, they know who they are.

# ABSTRACT

## SEMI-SUPERVISED LEARNING BASED NAMED ENTITY RECOGNITION FOR MORPHOLOGICALLY RICH LANGUAGES

In this study, we addressed the Named Entity Recognition (NER) problem for morphologically rich languages by employing a semi-supervised learning approach based on neural networks. We adopted a fast unsupervised method for learning continuous vector representations of words, and used these representations along with language independent features to develop a NER system. We evaluated our system for the highly inflectional Turkish and Czech languages and obtained better F-score performances than the previously published results for these languages. We improved the state-of-the-art F-score by 2.26% for Turkish and 1.53% for Czech. Unlike the previous state-of-the-art systems developed for these languages, our system does not make use of any language dependent features. Therefore, we believe it can easily be applied to other morphologically rich languages.

# ÖZET

## MORFOLOJİK AÇIDAN ZENGİN DİLLERDE YARI GÜDÜMLÜ ÖĞRENME TEKNİĞİYLE VARLIK İSMİ TANIMA

Bu çalışmamızda morfolojik açıdan zengin dillerde varlık ismi tanıma probleminin çözümüyle ilgilendik. Bu bağlamda, yapay sinir ağlarına dayalı yarı güdümlü öğrenme metodunu kullandık. İlk evrede, hızlı ve güdümsüz bir algoritma kullanarak kelimelerin çok boyutlu sürekli uzaydaki vektör gösterimlerini elde ettik. İkinci evrede ise, kelimelerin bu gösterimleri ile birlikte diğer bazı dil bağımsız öznitelikler de kullanarak varlık ismi tanıma sistemi geliştirdik. Oluşturduğumuz bu sistemi çok çekimli dillerden olan Türkçe ve Çekçe üzerinde denedik ve bu diller üzerinde yayınlanmış en gelişkin sistemlerden daha iyi performanslar elde ettik. Türkçe'de en gelişkin sistemi %2.26 ile, Çekçe'de ise en gelişkin sistemi %1.53 ile geliştirdik. Dile özgü öznitelikler de kullanan bu en gelişkin sistemlerden farklı olarak, çalışmamızda tamamen dilden bağımsız öznitelikler kullandık. Dolayısıyla yaptığımız bu çalışmanın morfolojik açıdan zengin olan diğer dillere de kolaylıkla ve başarıyla uygulanabileceğini düşünüyoruz.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| CNEC | Czech Named Entity Corpus |
| CoNLL | Conference on Computational Natural Language Learning |
| CRF | Conditional Random Fields |
| HMM | Hidden Markov Model |
| MUC | Message Understanding Conference |
| NER | Named Entity Recognition |
| NLP | Natural Language Processing |
| SVM | Support Vector Machines |

# 1. INTRODUCTION

Named Entity Recognition (NER) locates and classifies words in a given text into predefined categories. It is an important task prior to most further processes intended to extract a broader syntactic or semantic representation of the text [5]. To be more precise, NER is a constituent of many Natural Language Processing (NLP) tasks including information extraction, machine translation, question answering, and sentiment analysis. To illustrate, in the relation extraction sub-task of information extraction, the relations are defined between named entities (e.g. PERSON works in an ORGANIZATION). In the sentiment analysis field, the sentiment of a text can be attributed to named entities including companies and products. Likewise, in the question answering domain, the answers to questions are often named entities such as people, dates, etc. Due to being a crucial subtask of a number of NLP tasks, NER has been studied extensively by NLP researchers.

Named Entity Recognition was introduced at the $6^{th}$ Message Understanding Conference (MUC-6) as an understanding tasks. The task was to extract from a text all the words corresponding to seven categories grouped under three subtasks. These subtasks and the categories under them are described below [6]:

(i) ENAMEX: Proper names, acronyms, and other unique identifiers are considered in this subtask. These entities are categorized via the TYPE attribute as follows:
   - ORGANIZATION: named corporate, governmental, or other organizational entities
   - PERSON: named person or family
   - LOCATION: name of politically or geographically defined location (cities, provinces, countries, international regions, mountains, etc.)

(ii) TIMEX: Only temporal expressions are considered in this subtask. They are categorized via the TYPE attribute as follows:
   - DATE: partial or complete date expression
   - TIME: partial or complete expression of time of day

(iii) NUMEX: Numeric expressions are considered in this subtask. They can be money or percentages and may be expressed in either numeric or alphabetic form. The complete expression is covered in this task and is categorized via the TYPE attribute as follows:

- MONEY: monetary expression
- PERCENT: percentage

An example sentence with MUC-6 annotations taken from Grishman and Sundheim [1] is shown in Figure 1.1. Most of the NER studies have been inspired by the MUC-6 [1] and MUC-7 [7] evaluation programs. They have used the MUC categories and guidelines as a starting point. Compared to these original guidelines, the basic divergences in the studies that followed are the set of named entity categories used as well as the way the context of an entity is represented and utilized [5].

```
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 with MUC annotations taken from [1].

Conditional Random Fields (CRF) [8] are one of the most successful and widely used techniques for sequence labeling in several NLP tasks including NER [9, 10, 11]. However, recently, the neural network based semi-supervised learning approach has gained attention in the English NER studies [12, 13, 14, 15]. In the unsupervised stage of this approach, continuous vector representation of words are attained using a large amount of unlabeled data by employing a neural network. In the supervised stage, these feature vectors along with additional features are fed to another neural network to train a NER system. Since word representations constitute an important part of this approach, it is crucial to find good representations. The initial methods proposed to learn word representations have the drawback of large training durations with typical values of a few weeks [14, 15]. Recently, Mikolov *et al.* [3] showed that vector representations of words can be attained considerably faster, in a matter of hours, by employing a simpler neural network model. Although both representations

Table 1.1. Number of unique word forms in English and Turkish corpora of about 10M words [4].

| **Language** | **Vocabulary size** |
|---|---|
| **English** | 97,734 |
| **Turkish** | 474,957 |
| **Turkish (only roots)** | 94,235 |

are comparable, to our best knowledge, up until now, these representations have not been used for NER before.

Morphologically rich languages, such as Turkish, Czech and Hungarian, differ substantially from English since they have agglutinative or inflectional morphologies. In such languages production of hundreds of words from a given root is possible, which result in data sparsity problem. To illustrate, consider the following Turkish word *bul-un-ama-yabil-en*. It corresponds to *The one that may possibly not be found* in English. To be more concrete, using English and Turkish corpora of around 10 million words, Hakkani-Tür [4] showed that the vocabulary size for English is 97,734 and for Turkish is 474,957. However, when only root forms are considered, the vocabulary size for Turkish drops to 94,235, see Table 1.1. This analysis shows that 5 different Turkish word forms are generated from the same root on average. Due to this data sparsity problem, state-of-the-art systems for NER in morphologically rich languages usually make use of the analysis of the morphological structures of the languages and require language specific feature engineering [16, 17]. However, this makes them difficult to adapt to other languages.

In this study, we investigated using the neural network based semi-supervised learning approach for NER in morphologically rich languages. Unlike the previous semi-supervised NER studies, in the unsupervised stage for obtaining the word representations, we used the approach of Mikolov *et al.* [3]. In the supervised stage, in addition to these feature vectors, we benefited from additional language independent features such as word capitalization patterns, previous tag prediction, etc. We deliber-

Table 1.2. Overall F-score performance of the current state-of-the-art systems vs our system.

| | **Current state-of-the-art** | **Our system** |
|---|---|---|
| **Turkish** | 89.59% | 91.85% |
| **Czech** | 74.08% | 75.61% |

ately refrained from using any language specific features in order to make our system completely language independent and easily adaptable to other languages.

We evaluated the performance of our system on two highly inflectional morphologically rich languages, namely Turkish and Czech. We reported our results using the commonly accepted CoNLL metric [18]. The details of this metric are explained in the related works chapter. Finally, we compared our results with the state-of-the-art works for Turkish and Czech. The current state-of-the-art systems for Turkish and Czech NER are based on CRF and make use of language dependent features. The state-of-the-art result evaluated with respect to CoNLL metric for NER task in Turkish is reported by Şeker and Eryiğit [16] and has a 89.59% performance without using gazetteers. For the Czech language, it is reported by Konkol and Konopík [17] and has a 74.08% performance with gazetteers. Our system achieved an F-score performance of 91.85% for Turkish and 75.61% for Czech without using gazetteers and any language-specific features. These results set the new state-of-the-art performances for both Turkish and Czech NER. Table 1.2 shows the overall F-score performance of the current state-of-the-art systems and our system. The detailed results are presented in the experiments and results chapter.

The main contributions of our study can be summarized as follows: Firstly, we show that the neural network based semi-supervised learning approach that makes use of continuous word representations can be successfully applied to morphologically rich languages without performing any language specific morphological analysis. Secondly, we show that word representations obtained very fast by employing the approach of Mikolov *et al.* [3] are useful for NER. Finally, we outperform the current state-of-

the-art systems for Turkish and Czech NER without using any language dependent features.

The remainder of this thesis is organized as follows. The related work is discussed in the next chapter. The architecture of our system is described in Chapter 3. The data sets are given in Chapter 4. The experiments and results are presented in Chapter 5, and the thesis is concluded in Chapter 6.

# 2. RELATED WORK

In this chapter, we present the works related to our study. In the first section, we explain what the word representations are and where they are used in NLP. Then, we summarize the previous studies on Turkish and Czech NER in the second and third section, respectively.

## 2.1. Distributed word representations

Finding distributed representations has a long history [19, 20]. In [19], Hinton *et al.* describe distributed representations as follows: an entity is represented by a pattern of activity distributed over a number of computing elements such that each computing element is associated with representing many different entities. The power of this kind of representation is that it benefits from the processing abilities of networks of simple, neuron-like computing elements. Then, in the same year Rumelhart *et al.* [20] proposed a learning procedure called as back-propagation, which is very impressive and one of the most cited studies in the domain of neural networks. This procedure made possible to learn distributed representations effectively and is based on back-propagating errors in a layered neural network.

A neural network based architecture for building a probabilistic language model was proposed by Bengio *et al.* [21]. Their goal was to estimate the probability function for word sequences, expressed in terms of distributed representations. These representations are learnt by employing a feed-forward neural network consisting of input, projection, hidden and output layers. In this architecture, previous words to the current word are encoded at the input layer using 1-of-$V$ coding, where $V$ is the size of the vocabulary. The input layer is then projected to a projection layer using a shared projection matrix. Next, this projection, which is simply the concatenation of vectors of previous words, is passed through non-linear hidden layer and finally output of the hidden layer is fed to softmax layer to compute probability distribution over all the words in the vocabulary.

Collobert and Weston [12] showed that these distributed representation of words learnt from a language model are useful for a supervised neural network that aims to accomplish various NLP tasks including part-of-speech tagging, chunking, NER, and semantic role-labeling. In their later work, by implementing the same semi-supervised technique, Collobert *et al.* [15] achieved state-of-the-art performance results in several NLP tasks including NER. Their work improved the state-of-the-art accuracy for English NER from 89.31% to 89.59%. A neural network very similar to the one in [21] is employed in [12] and [15] except that the number of hidden layers are increased and a different cost function is used in the latter studies. One of the challenges for using this approach is the long training times of the deep neural network, which can take up to a few weeks, for obtaining the distributed representation of words.

In order to obtain continuous vector representations of words, Mikolov *et al.* [3] used a similar neural network to the feed-forward neural network employed in [21]. However, Mikolov *et al.* [3] removed the non-linear hidden layer in their architecture due to the fact that it is the layer that mainly causes the long training times. By employing this new architecture, distributed representations of words can be attained very fast (in a few hours with a computer having Intel core i7 processor and 8 GB RAM). Although a simpler model than [21] and [15] is used, the resulting word vectors are shown to perform better than [21] and [15] in a number of semantic and syntactic tasks [3].

## 2.2. Turkish NER

Morphologically rich languages pose challenges for several NLP tasks including NER. State-of-the-art systems developed for such languages usually depend on manually designed language specific features that utilize the rich morphological structures of the words. In this study, we propose using the semi-supervised neural network based approach for NER in morphologically rich languages without making use of any language dependent features. We applied the proposed system to two morphologically rich languages: Turkish and Czech.

Before giving the related works done on Turkish NER, let us explain the two metrics used commonly to evaluate the performances of the Turkish NER systems. These are as follows:

(i) MUC metric: In this metric, two F-scores are maintained for each class of entities, namely TYPE and TEXT scores. TYPE evaluates an assignment to be correct if the type of a named entity is assigned accurately without considering the boundary. In contrast to TYPE, TEXT considers the boundaries of the named entities and ignores their TYPEs. The overall TYPE and TEXT scores are micro-averaged F-score and the overall MUC score is the average of TYPE and TEXT scores.

(ii) CoNLL metric: Aside from Turkish NER studies, this metric is widely used in the recent studies for evaluating NER systems and defined in [18]. In our study, we also used this metric. The focus of this metric is to find phrase level named entities. That is, it evaluates an assignment to be correct if both the boundary and type of a named entity are assigned accurately. For computing the overall score micro-averaging is used in this metric as well.

To illustrate the scoring mechanism of these metrics and the difference between them, consider the example sentence with gold labels shown below in Figure 2.1.

```
  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 2.1. An example sentence with gold labels taken from [2].

Let's assume that this sentence is given to a NER system and the annotations produced by the system is illustrated in Figure 2.2. For the sake of simplicity, we will just show the computation of overall scores. Let's say the number of correct annotations predicted by the system is $C$, the total number of predictions done by the system is $T$, and the total number of entities in the gold data is $G$. Then the precision becomes $C/T$ and the recall becomes $C/G$.

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

Figure 2.2. Output of a system for the example sentence shown in Figure 2.1 [2].

For the MUC metric, $C$ is 4 (2 TYPE and 2 TEXT), $T$ is 10 (5 TYPE and 5 TEXT), and $G$ is 10 (5 TYPE and 5 TEXT). Therefore, the precision and the recall are $4/10 = 40\%$. Hence the overall F-score for MUC is also 40%. On the other hand, for the CoNLL metric, $C$ is 1 (only one of the predictions of the system matches exactly with the gold labels), $T$ is 5 (system makes 5 predictions), and $G$ is 5 (there are totally 5 entities in the gold data). Therefore, the precision and the recall are $1/5 = 20\%$. Hence the overall F-score is also 20%. After describing the evaluation metrics, now we can give the related studies done on Turkish NER.

One of the first studies on Turkish NER was conducted by Tür *et al.* [22]. They combined four models namely lexical, contextual, morphological and name tag models to estimate the named entity tag of a word. Firstly, the lexical information using only word tokens are captured for building the lexical model. Next, in the contextual model, the contextual information using the surrounding word tokens are captured. Then, the morphological parses of the words are used to construct a morphological model which captures the morphological information with respect to the corresponding case and name tag information. Lastly, the name tag model is employed to capture the name tag information. All of these models were based on Hidden Markov Model (HMM). They evaluated their system with respect to the MUC metrics [1, 7] and reported an F-score performance of 91.56% on the general news domain with ENAMEX type.

Tatar and Çiçekli [23] developed an automated rule learning system for named entity recognition in Turkish. Similar to [22], they exploited lexical, morphological, contextual and orthographic features in their study. As the lexical features, unlike the work in [22], they provided gazetteer information of words to the system. They used two level gazetteer hierarchy: the first level in the hierarchy corresponds to each

named entity class (e.g. Person, Location and so on) and the second level in the hierarchy details the gazetteer categorization (e.g. Location.Country, Location.City and so on). As the morphological features, morphological parses of the words are fed to the system. The information captured in the neighboring text of the named entities is used by the system as contextual features. As the orthographic features, they used the following four feature classes: Capitalization (Lower, Upper, Proper, Unclassified), Length Class (Short, Middle, Long), Length (the length of the token), and Type Class (Alpha, Numeric, Alphanumeric, Punctuation, Unclassified). They evaluated their system on a specific domain of text, namely on terrorism news, using the MUC metric with ENAMEX and TIMEX entity types and reported 91.08% F-score.

Yeniterzi employed word-level and morpheme-level models based on CRF [24]. The former model is the standard sequence of words representation, whilst the latter one represents morphological features of a word as states as well, in addition to its root. The word-level model looks at the word itself, its root form, its part-of-speech tag, whether it is proper-noun or not and whether its initial letter is uppercase or not. The morpheme-level model take the following features into account: the actual root of the word and the morphemes of the word, the part-of-speech tag of the root and the morphological tag of the morphemes, the proper-noun and the case feature of the word. Yeniterzi [24] evaluated these models using the CoNLL metric with the training set of the data set used in [22]. She reserved the 10% of this data for testing and used the rest for training her models. She obtained an F-score performance of 88.71% with word-level model and 88.94% with morpheme-level model.

Finally, the state-of-the-art work for Turkish NER was introduced by Şeker and Eryiğit [16]. They also employed CRF to build up the NER model by exploiting morphological, lexical, and gazetteer look-up features. As for the morphological features, the following are taken into account: stem of the word, part-of-speech tag of the word, all inflectional tags, the information that whether the word is nominal or not, and whether the word is proper noun or not. In addition to case features that give the information about lowercase and uppercase letters of the token, a binary feature is also used to indicate whether the current token is the beginning of a sentence or not as the

lexical features. Gazetteer look-up feature simply looks whether the current word is included in previously prepared gazetteer lists or not. The training and test sets used in their work are exactly the same with Yeniterzi [24]. They evaluated their system using both the MUC and CoNLL metrics and reported an F-score performance of 89.59% without gazetteers and 91.94% with gazetteers in CoNLL metric. They achieved an F-score of 92.83% without gazetteers and 94.59% with gazetteers in MUC metric.

### 2.3. Czech NER

Most work on Czech NER differs from traditional NER tasks because the most widely used corpus, which is the publicly available Czech Named Entity Corpus (CNEC) prepared by Ševcíkova *et al.* [25], is tagged in a hierarchical manner. In this type of annotation, a named entity belongs to a supertype and a type that results in different evaluation metrics. Below are descriptions of the three metrics used commonly in most of the Czech NER studies[17].

(i) Structured metric: In this metric, one can measure the performance of a system with any of the following three tasks:
   - Span recognition: In this task, the purpose is to find the spans of named entities. Recognizing their types is not tackled.
   - Supertype recognition: Here, the supertypes of named entities are considered and their spans are neglected.
   - Type recognition: Similar to supertype recognition, the span of an entity is neglected in this metric too. The motivation in this task is to find types of named entities.

(ii) Word by word metric: In this metric, each word is treated as a self-standing unit with supertype and type of it are considered together as one tag. When a named entity contains an embedded entity, the embedded entity takes the tag of the outer entity.

(iii) CoNLL metric: Although this metric was explained in the previous section, it is useful to describe it for Czech NER as well due to the hiearchical structure of the data set. In this case, similar to general usage, both the span and the supertype

should be correct. This means that if an entity consists of more than one word, type or supertype of an embedded entity is not important but the supertype of the phrase is considered to be important.

One of the first studies on Czech NER was done by Ševcíkova *et al.* [25]. As the learning algorithm, they used decision trees. To construct the NER model, they used a large set of features including capitalization pattern, lemma and part-of-speech tag of a token, contextual features that look at surrounding tokens, etc. They evaluated their system with CNEC corpus using structured metric and reported an F-score of 75% for span recognition, 68% for supertype recognition and 62% for type recognition tasks.

Kravalová and Žabokrtský [26] developed a NER system by employing support vector machines (SVM). They exploited morphological (part-of-speech tag, gender, case and number), capitalization, gazetteer, lexical (lemma) and contextual (mentioned features of preceding and succeeding tokens) features to build up the NER model. For the evaluation, they used structured metric and achieved an F-score of 76% for span recognition, 71% for supertype recognition and 68% for type recognition tasks on CNEC corpus.

Maximum entropy based NER for Czech is introduced by Konkol and Konopík [27]. They used a number of features similar to [26] to train the NER model such as morphological (part-of-speech tag, case, number and gender), lexical (lemma), capitalization, gazetteer, contextual (mentioned features of preceding and succeeding tokens) features and so on. They evaluated their system with CNEC corpus using word by word metric and reported an F-score of 72.94%.

Král [28] developed a NER system by employing CRF. He exploited a number of language independent and language dependent features to build the NER model. Language independent features include word itself, its prefixes and suffixes, word length, previous tag, prefixes and suffixes of neighbouring words and so on. As the language dependent features, gazetteer lists, lemma, part-of-speech tag, etc. are utilized. He

adopted a variant of CoNLL metric to evaluate his system on CNEC corpus and achieved an F-score of 58.4%.

Another maximum entropy based NER system is developed by Straková et al [29]. The classification features make use of the followings: form, lemma, part-of-speech tag of current word and neighbouring words in window $\pm 2$, orthographic features (capitalization, punctuation, lowercase and uppercase form of the word), prefixes and suffixes of length 4, regular expressions identifying possible year, date and time and gazetteer lists. For the evaluation, they used structured metric and reported an F-score of 86.83% for span recognition, 82.82% for supertype recognition and 79.23% for type recognition tasks on CNEC corpus.

Konkol and Konopík [17] build another NER system based on CRF. They utilized a number of language independent and language dependent features such as lemma, affixes, surrounding tokens, capitalization pattern, bi-gram language model, gazetteer lists, etc. Although, their system was designed to perform well in flat named entities, which corresponds to an evaluation with CoNLL metric, rather than hierarchical ones, they evaluated their system using all the metrics in order to be comparable with previous works. They achieved an F-score of 79% in supertype recognition task, 75.61% in word by word metric and 74.08% in CoNLL metric.

Among these studies, Straková *et al.* [29] and Konkol and Konopík [17] held state-of-the-art results. However, only Konkol and Konopík [17] evaluated their system according to the CoNLL metric. Therefore, we choose work of Konkol and Konopík [17] as the baseline in our study.

# 3. SEMI-SUPERVISED LEARNING BASED MODEL FOR NER

Our neural network based system consists of two stages. The first stage makes use of a huge amount of unlabeled data, whereas the second stage uses a rather restricted amount of labeled data. This kind of learning is called semi-supervised learning due to the fact that it makes use of both labeled and unlabeled data. The following subsections describe the details of our system.

## 3.1. Unsupervised stage

The main feature used by our NER model is the continuous word representations learned in the unsupervised stage. Therefore, it is important for our method to learn good vector representations of words, which map semantically similar words close to each other in the continuous vector space.

In order to obtain the continuous space vector representations of words, we used the publicly available implementation of Mikolov *et al.* [3], word2vec[1] , since it is much faster than the methods proposed by Bengio *et al.* [21] and Collobert *et al.* [15]. Most of the complexity in the work of Bengio *et al.* [21] and Collobert *et al.* [15] is introduced by the non-linear hidden layer in their models. Although this is what makes their models strong, it has the drawback of the long training times to obtain the vector representations of words, which restricts the amount of unlabeled data that can be used. Due to the fact that the non-linear hidden layer is removed and the projection layer is shared for all words in the architecture of Mikolov *et al.* [3], we were able to train our model with a huge amount of unlabeled data. Benefiting from large amounts of data is important, since as the amount of data increases, the obtained feature vectors of words become more representative [3].

Among the techniques described in Mikolov *et al.* [3], we used the continuous

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

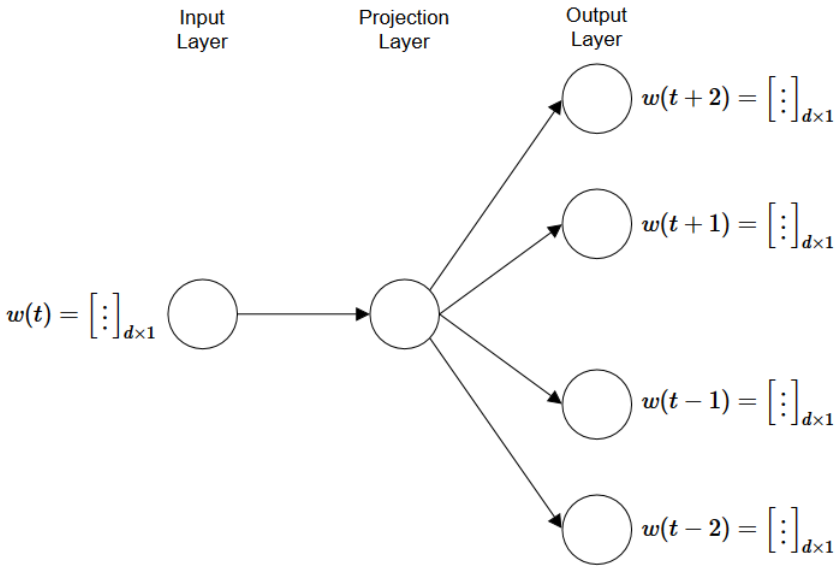


Figure 3.1. Architecture of the Skip-gram model that is employed to learn continuous vector representation of words [3]. Initially, words are mapped to random d-dimensional vectors. Then, each current word is fed to the log-linear classifier and representation of words that are neighbor of the current word are predicted.

Skip-gram model since it has been shown to be more successful at obtaining semantic representations of words [3]. In this architecture, the objective is to maximize the classification of a word according to the other words in the same sentence. Initially, words are mapped to random vectors of a specified dimension. Next, each current word is fed to a log-linear classifier with a projection layer. Then, representation of words that are neighbor of the current word are predicted. To be more formal, given a sequence of words, $w_1, w_2, \ldots, w_T$, the goal of the Skip-gram model is to maximize the average log probability

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

where $k$ is the size of the predefined range. Given a word, the inner summation computes the sum of the log probabilities of the previous and next $k$ words to it. The outer summation repeats this for all words. In the Skip-gram model, every word $w$ is associated with a learnable input and an output vector, $u_w$ and $v_w$ respectively. The

probability of word $w_i$ given word $w_j$ is defined as [30]

$$p(w_i|w_j) = \frac{\exp(u_{w_i}^T v_{w_j})}{\sum_{l=1}^{V} \exp(u_l^T v_{w_j})} \tag{3.2}$$

where $V$ is the number of words in the vocabulary. The cost of computing the gradient of this formula is expensive since it is proportional to the number of words in the vocabulary which is typically a large number. Therefore, instead of the full softmax, the hierarchical softmax [31] is employed that logarithmically reduces the complexity of computing $\log p(w_i|w_j)$. In the hierarchical softmax, the output layer is represented with a binary tree with the $V$ words as its leaves. It defines the probability of a word $w$ given word $w_g$ as follows [30]

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

where $\sigma(x)$ is the logistic function, $n(w, j)$ is the $j^{th}$ node on the path from the root to word $w$, $L(w)$ is the length of this path and $[\![x]\!]$ is 1 if $x$ is true and -1 otherwise. It can be shown that $\sum_{w=1}^{V} p(w|w_g) = 1$ [30]. In this formulation, every inner node $n$ of the tree has one representation $u_n$ and each word $w$ has one representation $v_w$. The cost of computing the gradient in this case is proportional to $L(w)$ whose upper bound is $\log V$. The model is trained using stochastic gradient descent and backpropagation is used for computing the gradient.

As the ranges or dimensions of vectors increase, the quality of the resulting vectors increases as well as the complexity. For our task, we chose the dimension as 200 and the range as 5, that is the representation of the previous and the next two words are predicted from the current token. The architecture we used is shown in Figure 3.1.

Using the Skip-gram architecture with a huge amount of unlabeled data in Turkish and Czech, we got the continuous space vector representations of words for both languages. By using the word2vec tool, we examined sample words and their closest neighbours in the vector space. We observed that the nearest neighbours are highly

related semantically to the queried words. Sample words in Turkish and their seven nearest neighbours in the vector space are shown in Table 3.1. The first words in columns 1 and 2 are person names in Turkish, and our model lists seven other person names as their nearest neighbours in the vector space. The first words in columns 3 and 4 are location names, "elazığ" is a city in Turkey and "ingiltere" (England) is a country name. The seven closest neighbours to "elazığ" are also cities in Turkey and the closest neighbors to "ingiltere" are also country names in Turkish. The first words of columns 5 and 6 are organization names, and organization entities are brought by our model as the nearest neighbours. Finally, the first word of the last column, "klarnet" is the name of a musical instrument (clarinet) and all of the seven closest neighbours to it are also names of musical instruments.

| **ahmet (name)** | **ayşe (name)** | **elazığ (city)** | **ingiltere (england)** | **huawei (org.)** | **dell (org.)** | **klarnet (clarinet)** |
|---|---|---|---|---|---|---|
| osman (name) | zeynep (name) | çorum (city) | italya (italy) | zte (org.) | toshiba (org.) | keman (violin) |
| mehmet (name) | necla (name) | erzurum (city) | almanya (germany) | ericsson (org.) | lenovo (org.) | çello (cello) |
| ismail (name) | zeliha (name) | mardin (city) | fransa (france) | ibm (org.) | nokia (org.) | trompet (trumpet) |
| ali (name) | hatice (name) | bitlis (city) | hollanda (holland) | cisco (org.) | samsung (org.) | viyolonsel (violoncello) |
| mustafa (name) | fatma (name) | yozgat (city) | belçika (belgium) | fujitsu (org.) | microsoft (org.) | kontrbas (contrabass) |
| cafer (name) | elif (name) | sivas (city) | ispanya (spain) | lenovo (org.) | apple (org.) | akordeon (accordion) |
| salih (name) | filiz (name) | gümüşhane (city) | isveç (sweden) | nokia (org.) | ibm (org.) | flüt (flute) |

Table 3.1. Example words in Turkish and their seven closest neighbors.

The results in Table 3.1 show that semantically similar words in natural language are placed close to each other in the vector space. This is a very useful feature, especially

for NER, where the semantic roles of words have important effects on distinguishing the named entity classes.

In addition to learning word representations, we also investigated the impact of incorporating word clustering to our NER system. We utilized the word2vec tool to cluster the resulting vector representations of the words using the k-means algorithm, and included the computed cluster ids of the words as additional features to our final NER system. Interestingly, these word vector clusters, which have not been used for NER before, led to improvement in performance.

## 3.2. Supervised stage

The supervised learning stage is where the NER models are formed. Since the features incorporated to form a model determine the quality of the resulting system, researchers tend to use language dependent features to increase the performances of their systems. The current state-of-the-art NER systems developed for morphologically rich languages are also usually based on manually designed features that utilize the language specific morphological analysis of the text. Although this approach usually improves the performance of a system due to the usage of linguistic knowledge, it is not portable and cannot be easily applied to different languages. In this study, we refrained from employing such engineered features. Instead, we restricted ourselves to only language independent features. In the following two subsections, we will explain the learning algorithm we employed and the features we exploited to construct our NER models.

### 3.2.1. Averaged Multiclass Perceptron

In order to train our models, we used the publicly available neural network implementation of Ratinov and Roth[32][2] , where the model and the features are specified with the LBJ [33]. In their work, they implemented a regularized averaged multiclass perceptron [34]. Let us now explain the algorithm briefly.

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

The basis of averaged multiclass perceptron is the classical perceptron algorithm invented by Rosenblatt [35]. Perceptron is a simple algorithm and allows for online learning, that is it can processes instances in the training set one at a time. The online perceptron algorithm initializes the weight vector, $w = [w_1, \ldots, w_d]^T$, to all zeros. The label of a new instance $x_t = [x_1, \ldots, x_d]^T$ is predicted to be $\hat{y}_t = sign(w^T x_t)$. If this prediction is false, the weight vector $w$ is updated to $w = w + \eta(y_t x_t)$, where $\eta$ is the learning rate. If the prediction is correct then $w$ stays as before. Then the same process is applied to the next example. Note that labels, $y_t$, are assumed to be in $\{-1, +1\}$. The algorithm is run again and again through the training set until a weight vector is found that satisfies a predefined stopping criteria.

Similar to other techniques for training linear classifiers, the perceptron algorithm can be used for multiclass classification as well [36]. Like classical perceptron, online learning can be performed in this case too. On each round $t$, the algorithm predicts a label out of $k$ possible labels from an instance vector $x_t \in \mathbb{R}^d$. In each run, the aim of the algorithm is obviously to minimize the number of prediction errors $M$, where:

$$M = \sum_{t=1}^{T} 1[\hat{y} \neq y] \quad . \tag{3.4}$$

Note that $1[P]$ is 1 if predicate $P$ is true and 0 otherwise. The prediction $\hat{y}$ is defined to be:

$$\hat{y} = \arg\max_{j \in [k]} (Wx)_j \quad , \tag{3.5}$$

where $W \in \mathbb{R}^{k \times d}$ is the weight matrix and $(Wx)_j$ is the $j^{th}$ element of the vector obtained by multiplying the weight matrix $W$ with the instance vector $x$.

The performance of weight matrix $W$ on an example $(x_t, y_t)$ is determined by checking whether $W$ predicts $y_t$ correctly or not. After each round, the algorithm may update its parameters in order to predict following instances more accurately. In the beginning, the multiclass perceptron initializes weight matrix to all zeros, that is

$W^1 = 0$. Then, it updates the weight matrix as follows:

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

where $U \in \mathbb{R}^{k \times d}$ is defined by

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

This means that if the algorithm predicts the label correctly, then no update is applied to weight matrix, $W$. However, if it predicts false, the weight matrix $W$ is updated as follows: $x_t$ is added to the $y_t^{th}$ row and subtracted from the $\hat{y}_t^{th}$ row of the $W$. After training, the weight matrix is stored and used to predict the label of instances during test. The prediction is again based on equation 3.5. The multiclass perceptron network is shown in Figure 3.2, where a $d$ dimensional input vector is assigned to one of $k$ possible classes.

In order to achieve better generalization, some modification to the classical perceptron algorithm is proposed in [34]. In the classical approach, only the last weight

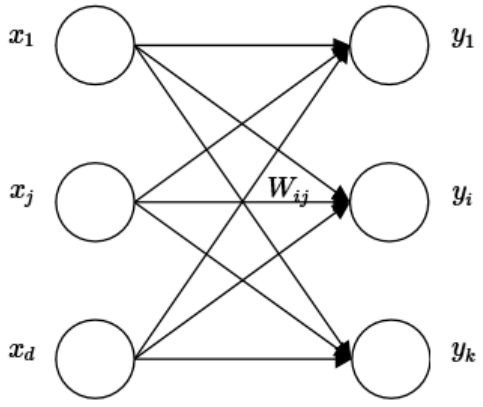


Figure 3.2. $k$-class perceptron where $x_j, j = 1, \ldots, d$ represent the inputs, $y_i, i = 1, \ldots, k$ represent the outputs and $W_{ij}$ represent the weight of the arrow from input $x_j$ to output $y_i$. Inputs are multiplied with the corresponding weights and then summed to give outputs. The maximum of outputs will be chosen to be the predicted class.

parameters are stored and used during tests. To see the drawback of this method, suppose a perceptron learns a good weight matrix during training and predicts training instances correctly. Thus, it does not update its weight parameters. However, suppose further that when it reaches the last instance of the training, it makes a false prediction and therefore updates its parameters. The update on this last example ruins the weight vectors of correct and estimated classes that have done well so far. But, it would be nice if previously found weight vectors were somehow not lost. In order to benefit from previous weight parameters, different approaches are proposed including voted and averaged perceptrons [34]. Since avereaged perceptron is used in our study, let us only give the details of averaged perceptron.

Averaged multiclass perceptron is very similar to standard multiclass perceptron. However, in this case, a weight matrix $W_{avg} \in \mathbb{R}^{k \times d}$ is also maintained. $W_{avg}$ is ini-

**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**

    $\hat{y} \Leftarrow \arg\max_j (W x_t)_j;$

    **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.

tialized to zero at first and after each iteration the weight matrix is added to it. After training, $W_{avg}$ is divided by the total number of iterations to give the average of all weight matrices found during training. $W_{avg}$ is then stored instead of the last weight matrix and used during tests. The procedure is shown in Figure 3.3 (Note that $W_k$ represents $k^{th}$ row of $W$). During training, after each epoch over the training set, we measured the performance of the model. When the performance stayed same for 10 epochs, we stopped the training. As the learning rate, we defined $\eta$ to be 0.1.

### 3.2.2. Training NER Models

In our framework, we did not change the architecture of Ratinov and Roth [32], but added some extra features. We exploited both local and non-local features to develop our models. These features are explained below.

(i) *Local features:* As the name suggests, features related to the neighbor of the current token, $x_i$, are considered in this case.
- *Context:* The tokens in the window of size two $c_i = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
- *Previous tags:* Named entity tag prediction of the previous three tokens
- *Type information:* Type information of the window $c_i$, i.e. is-capitalized, all-capitalized, all-digits, is-alphanumeric, and contains-apostrophe.
- *Prefixes:* First three and four characters of $x_i$ if it contains that many characters
- *Suffixes:* Last one, two, three and four characters of $x_i$ if it contains that many characters
- *Word representations:* Vector representation of each element in the window $c_i$
- *word2vec clusters:* We investigated the contribution of word2vec clusters and tried different numbers of clusters to obtain the best performance improvement. Finally, we found that 2000 and 256 clusters suit best for the Turkish and Czech NER tasks, respectively. We used the cluster id of each element in the window $c_i$.

(ii) *Non-local features:* In this case, sentence boundaries are ignored and global de-

pendencies are considered.

- *Context aggregation:* The tokens that are the same as the current token within a window of size 200 are investigated. The context features of each of these tokens are aggregated [32].
- *Extended prediction history:* Tag predictions of the tokens that are the same as the current token within the previous 1000 words are investigated. Then, the tag distribution of the token is used as a feature [32].

When using the lexical features, we normalized the numbers as in the work of Turian *et al.* [14]. To illustrate, 2014 is represented as *DDDD* and (0212) 153 69 74 is represented as (*DDDD*) *DDD* *DD* *DD*. This normalization enables us to achieve a degree of abstraction to numeric expressions such as years and phone numbers.

Besides the features used, the representation of the named entities also affects the performance of a NER system. Among the alternative encoding schemes, such as BIO and BILOU, we used BILOU as the representation scheme, since it has been shown to perform better than the BIO representation [32].

In the BILOU representation scheme, see Figure 3.4, a named entity that has multiple tokens is encoded as **B**eginning, **I**nside and **L**ast, and as **U**nit if it has one token. If the token is not a named entity it is encoded as **O**utside of any type of named entity. During testing, after tagging with respect to the BILOU scheme, the tags are

```
Mr.(O) Dooner(PERSON_U) met(O) with(O) Martin(PERSON_B) Puris(PERSON_L) ,(O) president(O)
and(O) chief(O) executive(O) officer(O) of(O) Ammirati(ORGANIZATION_B) &(ORGANIZATION_I)
Puris(ORGANIZATION_L) ,(O) about(O) McCann(ORGANIZATION_U)'s(O) acquiring(O) the(O) agency(O)
with(O) billings(O) of(O) $400(MONEY_B) million(MONEY_L) ,(O) but(O) nothing(O) has(O)
materialized(O) .(O)
```

Figure 3.4. Annotation of the example sentence shown in Figure 1.1 with BILOU representation. The named entity tag category of each token is indicated inside parentheses.

```
Mr.(O) Dooner(PERSON_B) met(O) with(O) Martin(PERSON_B) Puris(PERSON_I) ,(O) president(O)
and(O) chief(O) executive(O) officer(O) of(O) Ammirati(ORGANIZATION_B) &(ORGANIZATION_I)
Puris(ORGANIZATION_I) ,(O) about(O) McCann(ORGANIZATION_B)'s(O) acquiring(O) the(O) agency(O)
with(O) billings(O) of(O) $400(MONEY_B) million(MONEY_I) ,(O) but(O) nothing(O) has(O)
materialized(O) .(O)
```

Figure 3.5. Annotation of the example sentence shown in Figure 1.1 with BIO representation. The named entity tag category of each token is indicated inside parentheses.

converted to BIO tags, see Figure 3.5. That is, the tokens tagged as U and L are changed to B and I, respectively. This is required for using the standard performance evaluation method of CoNLL.

To be more concrete, let us give an example sentence and show the features we used on it. Assume that we are going to label the following sentence: *"Boğaziçi University is located in İstanbul."* and there are only three categories of entities, namely person, organization and location. Assume further that the system has already made predictions for the first three words, *B-ORG* for *Boğaziçi*, *L-ORG* for *University* and *O* for *is*. Now, the system is going to predict the tag of *located*. Let's call the current word as $x_t$, its word representation as $w_t = [w_1, \ldots, w_d]^T$, its cluster id as $i_t$ and its tag as $y_t$. Then each feature for *"located"*, $x_t$, will be as follows:

- *Context:* ($x_{t-2}$: university, $x_{t-1}$: is, $x_t$: located, $x_{t+1}$: in, $x_{t+2}$: istanbul)
- *Previous tags:* ($y_{t-3}$: B-ORG, $y_{t-2}$: L-ORG, $y_{t-1}$: O)
- *Type information:* (is-capitalized: false, all-capitalized: false, all-digits: false, is-alphanumeric: true, contains-apostrophe: false)
- *Prefixes:* (first three character of $x_t$: loc, first four character of $x_t$: loca)
- *Suffixes:* (last one character of $x_t$: d, last two character of $x_t$: ed, last three character of $x_t$: ted, last four character of $x_t$: ated)
- *Word representations:* ($w_{t-2}$, $w_{t-1}$, $w_t$, $w_{t+1}$, $w_{t+2}$)
- *word2vec clusters:* ($i_{t-2}$, $i_{t-1}$, $i_t$, $i_{t+1}$, $i_{t+2}$)
- *Context aggregation:* Assume there is only one same word with $x_t$ in the window

of size 200 and it passes 15 tokens after it in the following sentence: *İstanbul is located on two continents.* So, the feature vector is ($x_{t+13}$: istanbul, $x_{t+14}$: is, $x_{t+15}$:located, $x_{t+16}$: on, $x_{t+17}$: two)

- *Extended prediction history:* The tag distribution of $x_t$ by searching $x_t$ in previous 1000 tokens, assume it is computed to be (PER: 0.1, ORG: 0.1, LOC: 0.1, O: 0.7)

These feature vectors are concatenated and fed to the averaged perceptron to predict the named entity tag of *"located"* as shown in Figure 3.2. Note that discrete variables with three or more possible values are encoded using 1-of-N encoding. To illustrate, let's say we have only 4 alternatives for a type of feature that are *usb*, *mouse*, *keyboard* and *monitor*. Then, their encoding will be as follows: 1000, 0100, 0010 and 0001. However, since the system may encounter an unknown feature during testing, which it has not seen in the training, we need to consider an encoding for unknown features as well. Therefore, the final encoding will be: 10000, 01000, 00100, 00010 and 00001, where the first four ones are for the known features and the last one is for unknowns.

The performance of our model and the contribution of each feature to the model are presented in the experiments and results chapter.

# 4. DATA SETS

In this section, we provide the details of the unlabeled and labeled data sets that we used for training and testing our system.

## 4.1. Turkish data sets

In the unsupervised stage, we used data collected from several Turkish news sites. We tokenized the data by using the publicly available zemberek[3] tool and then lowercased it. By lowercasing, we aimed to limit the number of words. The data that we used in this stage contain 63.72M sentences that correspond to a total of 1.02B words and 1.36M unique words.

In the supervised stage, we used the data set prepared by Tür *et al.* [22] which is the most commonly used one for evaluating Turkish NER systems including the state-of-the-art system proposed by Şeker and Eryiğit [16]. It is partitioned into training and test sets, that contain 450K words and 50K words, respectively. This data set is annotated according to the ENAMEX tags, that is, it contains person (PER), location (LOC) and organization (ORG) entities. An example sentence from the data set is shown below (It means: *Brazilian artist Tania Maria, who is a vocal and piano virtuoso, will give the festival's opening concert in İstanbul*).

```
Festival 'in <b_enamex TYPE="LOCATION">Istanbul<e_enamex> 'daki
açılış konserini vokal ve piyano virtüözü Brezilyalı sanatçı <b_enamex
TYPE="PERSON">Tania Maria<e_enamex> yapacak
```

Number of entities in the training and test sets are shown in Table 4.1. It is worth noting that no matter an entity consists of only one token or more, it is counted as one since the CoNLL evaluation task consider phrases and not tokens.

[3]https://github.com/ahmetaa/zemberek-nlp

Table 4.1. Number of entities in the Turkish data set.

| | **Train** | **Test** |
|---|---|---|
| **PER** | 14481 | 1594 |
| **ORG** | 9034 | 864 |
| **LOC** | 9409 | 1090 |

## 4.2. Czech data sets

For the unlabeled data, we used the publicly available data crawled from Czech news sites provided by the ACL machine translation workshop[4] . We tokenized the data using the Moses tokenizer[5] and then applied lowercasing. This data set contains 36.42M sentences corresponding to 635.99M words and 906K unique words.

While training and testing our Czech NER system in the supervised stage, we used the CNEC 1.1 data set prepared by Ševcíková *et al.* [25]. It is divided into training, development, and test sets, which contain 124K, 15K and 15K tokens, respectively. The number of entities in these sets are shown in Table 4.2. Unlike the traditional tagging schemes, CNEC is annotated by using two level hierarchical named entities. The first level is named as *supertype* and the second level is named as *type*. An example sentence from the data set is shown below (It means: *It's a bitter disappointment and warning for our hockey, but the misery continued even in the duel with Devils Milan.* ).

```
Pro náš hokej trpké poznání a výstraha , ale trápení pokračovalo i
v souboji s <ic Devils <gu Milán>> .
```

The first character of a tag determines the supertype of a named entity and the second character determines its type. In the example sentence, "i" tells us that "Devils Milán" is an institution name, "c" tells us that "Devils" is a cultural/educational/scientific in-

[4]http://www.statmt.org/wmt14/translation-task.html
[5]https://github.com/moses-smt/mosesdecoder/tree/master/scripts/tokenizer

Table 4.2. Number of entities in the CNEC 1.1 data set.

| | **Train** | **Dev.** | **Test** |
|---|---|---|---|
| **Addr.** | 109 | 23 | 14 |
| **Geo.** | 2890 | 399 | 340 |
| **Inst.** | 2595 | 322 | 309 |
| **Media** | 244 | 34 | 32 |
| **Person** | 3704 | 497 | 472 |
| **Time** | 2384 | 275 | 361 |
| **Other** | 2432 | 321 | 378 |

stitution, “g” tells us that “Milán” is a geographical name, and lastly “u” tells us that “Milán” is a castle/chateau. Although this type of tagging is much more informative than the traditional ones, it leads to different types of evaluation approaches. Therefore, we used the transformed version of this data set prepared by Konkol and Konopík [17]. The original corpus used 10 supertypes and 62 types, whilst the transformed corpus uses only 7 supertypes. These are as follows: address (Addr.), geography (Geo.), institution (Inst.), media, person, time and other (for miscellaneous named entities). This transformation aims to make the data set compatible with the CoNLL evaluation metric, so that the results become comparable with other systems.

# 5. EXPERIMENTS AND RESULTS

In this chapter, we present the performance of our system and compare the results with the state-of-the-art studies. For the sake of demonstration of our final system, we prepared a simple web interface where a user can enter a text and see the named entities in the text found by our system[6] . A screenshot from the web interface with an example Turkish sentence tagged by our system is shown in Figure 5.1.

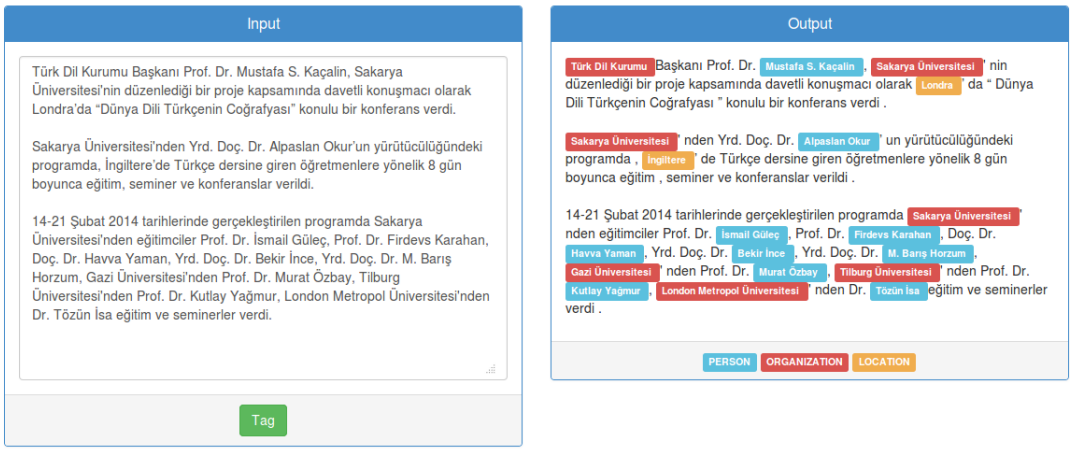


Figure 5.1. An example Turkish text tagged by our system on our simple web interface. Blue, yellow and red colors are used for person, organization, and location entities, respectively.

In the evaluation phase, we first trained our word representations. Then, we trained our NER models using these word representations. The first stage took around 1 hour for Turkish words and 30 minutes for Czech words due to the different sizes of the corpora for these languages. In the second stage, training a NER model for a language took around 7 hours. All experiments were performed with a computer having 16 GB RAM and Intel Core i7 processor.

In order to explore the contribution of each feature we used, we trained six different models for each language. We chose *context features* to be the base and added each feature to it. The results are evaluated with respect to the CoNLL metric and shown in

[6]We thank Erdem Orman for his contributions to the implemention of the web interface.

Table 5.1. The first row corresponds to the results obtained when only the *context features* are used in Turkish NER. Each successive row shows the performance of the corresponding feature combined with the *context features*.

| **Features** | **PER** | **ORG** | **LOC** | **Overall** |
|---|---|---|---|---|
| Context features | 81.65 | 74.96 | 88.43 | 82.21 |
| Previous tags | 84.84 | 78.86 | 89.23 | 84.84 |
| Word type | 88.49 | 78.85 | 89.32 | 86.43 |
| Affixes | 83.04 | 76.82 | 87.96 | 83.10 |
| Word represent. | 91.24 | 79.95 | 90.50 | 88.28 |
| word2vec clusters | 88.73 | 77.03 | 89.43 | 86.15 |

Table 5.2. The first row corresponds to the results obtained when only the *context features* are used in Czech NER. Each successive row shows the performance of the corresponding feature combined with the *context features*.

| **Features** | **Addr.** | **Geo.** | **Inst.** | **Media** | **Other** | **Person** | **Time** | **Overall** |
|---|---|---|---|---|---|---|---|---|
| Context features | 22.22 | 56.18 | 29.88 | 34.04 | 50.14 | 40.59 | 86.42 | 53.70 |
| + Previous tags | 33.33 | 56.51 | 36.65 | 37.50 | 58.13 | 45.31 | 88.24 | 58.36 |
| + Word type | 25.00 | 61.04 | 37.44 | 40.82 | 48.78 | 61.69 | 87.34 | 59.84 |
| + Affixes | 23.53 | 61.49 | 33.52 | 36.74 | 49.59 | 54.39 | 85.60 | 57.45 |
| + Word represent. | 35.29 | 69.25 | 46.26 | 42.31 | 51.58 | 66.25 | 88.98 | 64.72 |
| + word2vec clusters | 22.22 | 64.07 | 37.62 | 42.31 | 51.25 | 58.23 | 88.46 | 60.53 |

Table 5.1 and Table 5.2. The first rows in the tables correspond to the results obtained when only the *context features* are used. Each successive row shows the performance of the corresponding feature combined with the base feature. These experiments show that the order of importance of each feature to NER performance seems to be parallel. We think that this is because both languages are similar in terms of their morphology. The results indicate that the *word representation* feature contributes most and the *word2vec clusters* feature, which has not been tried for NER before, contributes remarkably.

We also examined the cumulative contribution of features. The results are shown in Table 5.3 and Table 5.4. The first rows in the tables correspond to the results obtained when only the *context features* are used. Each successive row shows the

Table 5.3. The first row corresponds to the results obtained when only the *context features* are used in Turkish NER. Each successive row shows the performance obtained after including the corresponding feature to the model cumulatively.

| **Features** | **PER** | **ORG** | **LOC** | **Overall** |
|---|---|---|---|---|
| Context features | 81.65 | 74.96 | 88.43 | 82.21 |
| + Previous tags | 84.84 | 78.86 | 89.23 | 84.84 |
| + Word type | 91.45 | 82.45 | 90.17 | 88.94 |
| + Affixes | 92.26 | 83.53 | 90.73 | 89.73 |
| + Word represent. | 94.36 | 85.51 | 92.61 | 91.71 |
| + word2vec clusters | 94.69 | 85.78 | 92.40 | **91.85** |

Table 5.4. The first row corresponds to the results obtained when only the *context features* are used in Czech NER. Each successive row shows the performance obtained after including the corresponding feature to the model cumulatively.

| **Features** | **Addr.** | **Geo.** | **Inst.** | **Media** | **Other** | **Person** | **Time** | **Overall** |
|---|---|---|---|---|---|---|---|---|
| Context features | 22.22 | 56.18 | 29.88 | 34.04 | 50.14 | 40.59 | 86.42 | 53.70 |
| + Previous tags | 33.33 | 56.51 | 36.65 | 37.50 | 58.13 | 45.31 | 88.24 | 58.36 |
| + Word type | 44.44 | 61.13 | 44.52 | 47.83 | 60.09 | 65.59 | 88.86 | 64.68 |
| + Affixes | 44.44 | 67.14 | 48.00 | 44.00 | 62.99 | 70.31 | 90.27 | 68.38 |
| + Word represent. | 33.33 | 76.77 | 64.18 | 53.85 | 64.82 | 80.36 | 89.90 | 75.52 |
| + word2vec clusters | 33.33 | 76.03 | 62.86 | 54.90 | 65.81 | 81.39 | 89.96 | **75.61** |

performance obtained after including the corresponding feature to the model in the previous row. Since *previous tag*, *word type* and *affix* features are disjoint, they contribute as much as they did to the base feature. However, this is not the same for *word representation* feature. This is because it learns a part of these features as well.

The comparison between our system and the state-of-the-art system for Turkish [16] is given in Table 5.5. In Şeker and Eryiğit [16], CRF is employed as the learning algorithm. In addition to some language independent features, a number of language dependent features are also used. To be more precise, the stems of words, their part of speech tags, all inflectional features of the tokens, and information whether a token is a proper noun or not, are used. Including these language dependent features resulted in an F-measure of 89.59% in CoNLL metric without using gazetteers, and an F-measure

Table 5.5. Comparison of our system with the state-of-the-art Turkish NER.

| **System** | **PER** | **ORG** | **LOC** | **Overall** |
|---|---|---|---|---|
| Şeker and Eryiğit, (2012) without gazetteer | 90.65 | 86.12 | 90.74 | 89.59 |
| Our final system | 94.69 | 85.78 | 92.40 | **91.85** |

Table 5.6. Comparison of our system with the state-of-the-art Czech NER.

| **System** | **Addr.** | **Geo.** | **Inst.** | **Media** | **Other** | **Person** | **Time** | **Overall** |
|---|---|---|---|---|---|---|---|---|
| Konkol and Konopík, (2013) | 58.33 | 77.37 | 67.02 | 39.13 | 55.96 | 82.29 | 86.68 | 74.08 |
| Our final system | 33.33 | 76.03 | 62.86 | 54.90 | 65.81 | 81.39 | 89.96 | **75.61** |

performance of 91.94% with gazetteers. Gazetteers are simply sets of lists containing names of entities such as people, locations, organisations, time expressions and so on. They are shown to increase performance of NER systems and exploited as follows: Each gazetteer list is designed to contain one type of entities. Next, boolean features indicating whether the word is found in each of these lists or not are attained [15]. Then, gazetteer features along with other features are used to develop a NER model. While predicting the named entity tag of a word, it is possible to use gazetteer feature of previous and next words to the current word as well [14]. Our system outperforms these results without using any language dependent features, when gazetteers are not included[7] .

We also compared our system with the state-of-the-art Czech NER system [17]. The comparison is shown in Table 5.6. In fact, both Straková *et al.* [29] and Konkol and Konopík [17] hold state-of-the-art results for Czech NER. However, only Konkol and Konopík [17] evaluate their system according to the CoNLL metric. Therefore, we were able to compare our system with theirs. They report an F-score performance of 74.08% with gazetteers. As the learning algorithm, they used CRF. Their approach includes some language dependent features such as word lemmas obtained by language

[7]We were not able to make a comparison of our system with the gazetteer feature added, since the gazetteers used in Şeker and Eryiğit [16] are not publicly available.

specific morphological analysis and gazetteer lists. It is worth noting that our system does not use any gazetteers and still outperforms their approach with an F-score of 75.61%.

The number of entities in the training sets are not distributed uniformly, see Table 4.1 and Table 4.2. Analyzing the training set sizes for each tag suggests that the performance of our system is relatively lower when there is less training data, as expected. For instance, there are only 109 *Address* tags in the Czech training set but over 2000 *Time* tags. This is one possible reason why *Address* tag perform worse compared to *Time* tag. In addition to this, the test sizes for the *Address* and *Media* tags are 14 and 32 respectively, which make their performance evaluation fragile. Therefore, results obtained for these classes of entities may possibly not represent the quality of the model.

# 6. CONCLUSIONS AND FUTURE WORKS

In this study, we investigated using semi-supervised learning based on distributed word representations and neural networks for NER in morphologically rich languages. First, we learned continuous space vector representations of words from a huge amount of unlabeled data collected from a number of news sites. Then, by using these vector representations of words and additional features extracted from the data, we trained an averaged perceptron using labeled data sets to learn NER models for Turkish and Czech, which are highly inflectional morphologically rich languages. Finally, we evaluated our method using the CoNNL metrics and compared our results with the state-of-the-art systems proposed for Turkish and Czech, which make use of language-specific morphological analysis. We showed that utilizing the continuous vector space representations of words in a semi-supervised setting is a powerful approach for NER, and can result in state-of-the-art performance without using any language dependent features for morphologically rich languages. Therefore, we believe, this approach can be easily applied to other languages.

The task of sentiment analysis draws high attention both from academia and industry. Recognizing named entities is an important subtask of sentiment analysis since the sentiment of a text can be attributed to companies, organizations or people. Therefore, adapting the proposed system to the social media domain and evaluating its performance, e.g. on Twitter data, would be a nice extension to this study.

In this study, we restricted ourselves to consider language independent features only. However, language dependent features may be exploited as well. Thus, a potential improvement to this study would be to investigate whether the performance of the proposed system can be further improved by incorporating language-specific features and gazetteers.

Although the perceptron algorithm is old and simple, it has proven to be successful in many practical problems [36]. However, its decision boundaries can only be

linear. This restricts the algorithm to only linearly solvable problems. Therefore, another potential direction for improvement would be to employ a multilayer perceptron in the supervised stage. Multilayer perceptron is a modification of the standard linear perceptron and can distinguish data that are not linearly separable. Hence, it may result in better performances than the linear perceptron used in this study.

# REFERENCES


1. 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.

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

3. 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.

4. Hakkani-Tür, D. Z., *Statistical Language Modelling for Turkish, Ph.D. thesis*, Department of Computer Engineering, Bilkent University, 2000.

5. Tur, G. and R. De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, Wiley, New York, 2011.

6. 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.

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

8. Lafferty, J. D., A. McCallum and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

9. McCallum, A. and W. Li, "Early Results for Named Entity Recognition with Con-

ditional Random Fields, Feature Induction and Web-enhanced Lexicons", *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) 2003 - Volume 4*, CONLL '03, pp. 188–191, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003.

10. Finkel, J. R., T. Grenager and C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling", *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pp. 363–370, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005.

11. Ekbal, A. and S. Bandyopadhyay, "A Conditional Random Field Approach for Named Entity Recognition in Bengali and Hindi", *Linguistic Issues in Language Technology*, Vol. 2, No. 1, 2009.

12. 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.

13. Turian, J., Y. Bengio, L. Ratinov and D. Roth, "A Preliminary Evaluation of Word Representations for Named-entity Recognition", *Neural Information Processing Systems (NIPS) Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009.

14. Turian, J., L. Ratinov and Y. Bengio, "Word Representations: A Simple and General Method for Semi-supervised Learning", *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pp. 384–394, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010.

15. 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.

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

17. Konkol, M. and M. Konopík, "CRF-Based Czech Named Entity Recognizer and Consolidation of Czech NER Research", *Text, Speech and Dialogue (TSD)*, pp. 153–160, 2013.

18. 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) 2003 - Volume 4*, CONLL '03, pp. 142–147, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003.

19. Hinton, G. E., J. L. McClelland and D. E. Rumelhart, "Distributed Representations", *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, pp. 77–109, MIT Press, 1986.

20. 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.

21. 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.

22. 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.

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

24. 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.

25. Ševčíková, M., Z. Žabokrtsky and O. Krůza, "Named Entities in Czech: Annotating Data and Developing NE Tagger", *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, TSD'07, pp. 188–195, Springer-Verlag, Berlin, Heidelberg, 2007.

26. Kravalová, J. and Z. Žabokrtský, "Czech Named Entity Corpus and SVM-based Recognizer", *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, NEWS '09, pp. 194–201, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009.

27. Konkol, M. and M. Konopík, "Maximum Entropy Named Entity Recognition for Czech Language", *Proceedings of the 14th International Conference on Text, Speech and Dialogue*, TSD'11, pp. 203–210, Springer-Verlag, Berlin, Heidelberg, 2011.

28. Král, P., "Features for Named Entity Recognition in Czech Language", *Knowledge Engineering and Ontology Development (KEOD)*, pp. 437–441, 2011.

29. Straková, J., M. Straka and J. Hajic, "A New State-of-The-Art Czech Named Entity Recognizer.", *Text, Speech and Dialogue (TSD)*, pp. 68–75, 2013.

30. 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.

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

32. 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.

33. Rizzolo, N. and D. Roth, "Modeling Discriminative Global Inference", *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pp. 597–604, IEEE Computer Society, Washington, DC, USA, 2007.

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

35. 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.

36. 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.