Named Entity Recognition for Social Media Text

Yaxi Zhang

Uppsala University
Department of Linguistics and Philology
Master Programme in Language Technology
Master’s Thesis in Language Technology, 30 ECTS credits
October 28, 2019

Supervisors:
Ali Basirat, Uppsala University
Abstract

This thesis aims to perform named entity recognition for English social media texts. Named Entity Recognition (NER) is applied in many NLP tasks as an important preprocessing procedure. Social media texts contain lots of real-time data and therefore serve as a valuable source for information extraction. Nevertheless, NER for social media texts is a rather challenging task due to the noisy context. Traditional approaches to deal with this task use hand-crafted features but prove to be both time-consuming and very task-specific. As a result, they fail to deliver satisfactory performance. The goal of this thesis is to tackle this task by automatically identifying and annotating the named entities with multiple types with the help of neural network methods.

In this thesis, we experiment with three different word embeddings and character embedding neural network architectures which combine long short-term memory (LSTM), bidirectional LSTM (BI-LSTM) and conditional random field (CRF) to get the best result. The data and evaluation tool come from the previous shared tasks on Noisy User-generated Text (W-NUT) in 2017. We achieve the best F1 score 42.44 using BI-LSTM-CRF with character-level representation extracted by a BI-LSTM, and pre-trained word embeddings trained by GloVe (Pennington et al., 2014). We also find out that the results could be improved with larger training data sets.
## Contents

Preface

1 Introduction 5
1.1 Purpose 5
1.2 Outline 5

2 Background 7
2.1 Natural language processing for social media text 7
2.2 Named entity recognition 8
   2.2.1 Rule-based approaches 8
   2.2.2 Data-driven approaches 9
   2.2.3 Neural network approaches 10
2.3 Shared tasks for NER on social media text 10
2.4 Neural network model for NER tasks 11
   2.4.1 LSTM 11
   2.4.2 Bidirectional LSTM 13
   2.4.3 Conditional Random Field 13

3 Data and experimental setup 15
3.1 Data and evaluation 15
   3.1.1 Data 15
   3.1.2 Word embeddings 16
   3.1.3 Evaluation 16
3.2 Experimental setup 16
   3.2.1 BI-LSTM for character-level representation 17
   3.2.2 CNN for character-level representation 17
   3.2.3 Combined character-level and word-level representation 18

4 Experiment results and discussion 20
4.1 Evaluate between three models 20
4.2 Hyper-parameter optimization 21
4.3 Final test result 24
4.4 Error analysis 25

5 Conclusion and future work 27
Preface

I want to thank my supervisor Ali Basirat for his kind help and valuable advice. I would also like to thank my family, friends, all the teachers and my classmates for their support during the past two years of my master studies.
1 Introduction

A named entity is a real-world object such as a person (Mark Twain), a location (New York) or a product (iPhone). Named Entity Recognition (NER) is a NLP task which identifies various named entities that appear in the text. NER is one of the NLP tasks and is used in many information extraction pipelines as an important part. Social media contain lots of real-time data, and thus is valuable for information extraction. NER for social media text is still a very challenging task. Most of the NER tools perform badly on social media text since it contains various topics and is considered to be “noisy” for using hashtags and emoji. In order to compare the performance of different NER tools and methods on social media text, and also to encourage more effort and research in this area, a shared task was organized as part of the Workshop on Noisy User-generated Text (W-NUT) focusing on named entity recognition task (Baldwin et al., 2015).

1.1 Purpose

This thesis intends to solve the task that was in line with the W-NUT shared task in 2017 (Derczynski et al., 2017), which focused more on identifying entities that are unusual and have not been seen before. New named entities are coming up continuously while some of the current existing named entities are not used anymore or replaced by the new ones. Thus it is essential for the NER tools to have the ability to recognize previously unseen entities.

NER task is a typical sequence labeling task and one of the dominant approaches of handling sequence labeling task is Conditional Random Fields (CRF) (Lafferty et al., 2001). However, a traditional CRF model relies heavily on hand-crafted features, which is time-consuming and hard to develop. Also, the hand-crafted features built for a specific domain are usually hard to be used in another domain, while social media text contains various topics. Therefore, the traditional CRF method is not suitable for this task. In this thesis, we aim to use neural network models to automatically generate features to feed into a CRF model and try to find the most suitable model for this task.

1.2 Outline

The rest of the thesis is structured as follows:

- Chapter 2 introduces some background and related work on NER in general, and the development of NER on social media text.
- Chapter 3 focuses on the description of the neural networks that are used in this thesis. We also explain the neural architecture to extract character-level information.
• Chapter 4 elaborates on the data and evaluation tool of the experiments. We evaluate different models and optimize the hyper-parameters on the best-performed system based on the development data, and then apply it to the test data to obtain the final experimental results.

• Chapter 5 gives the conclusion of this thesis and makes suggestions for future work.
2 Background

To begin with, this chapter studies the general natural language processing research on social media data. Secondly, we introduce the background and related works of named entity recognition. Furthermore, we will have a brief overview of the recently held shared tasks for NER on social media text. Last but not least, a brief introduction of the dominant neural method of handling NER tasks has been presented.

2.1 Natural language processing for social media text

Social media resources have drawn more and more attention from NLP researchers in recent years. Social media contain various topics and discussions in real-time, together with user information and geolocation information, which inspired scholars to conduct different kinds of exciting research, for example, insights into public health (Paul and Dredze, 2011), political polarization (Conover et al., 2011), trends prediction (Chakraborty et al., 2017) and earthquake detection (Sakaki et al., 2010).

On the other hand, being "noisy" (Preotiuc-Pietro et al., 2012; Yin et al., 2015) makes social network text hard to use. Firstly, the social media text often contains hashtags, emoji, URL or other non-natural languages like code snippet. Secondly, many social media platforms like Facebook or Twitter have a length limit for user inputs, plus the diversity of topics, make the messages and posts very short and thus lack of context. Thirdly, in order to meet the character limitation, people tend to use deviations from the standard language (u = you, 4 = for) or omit apostrophe (dont = don’t) to save time. People also use long vowels to express emphasis (soooooo goooolllld). A standard way of dealing with the noise is to try to remove non-standard words and unknown words. Therefore, text normalization is usually performed as the first step (Sproat et al., 2001) when dealing with social media text in NLP tasks.

There is a great amount of NLP research on social media text. POS tagging task is one of the most fundamental parts of the linguistic pipeline. The POS tagging task mainly focuses on how to tag noisy tokens. Gimpel et al. (2010) proposed to use clustering to deal with the noise tokens trained from human-annotated tweets. Darling et al. (2012) used lexicon features through topic models and Brown clustering (Clark, 2003) to handle noise tokens.

The NER task on social media text also focuses on how to deal with the noise text to improve results. Toh et al. (2015) proposed to use Brown clustering and the K-means algorithm to generate word representations as word cluster features. Yamada et al. (2015) tried to detect entities and confirmed the detection by searching through knowledge bases such as Wikipedia.

To sum up, different types of interesting research have been conducted on social media texts as well as basic NLP tasks like POS tagging and NER. All of
these research has been mainly focused on how to handle the noise of the text in order to achieve good performance.

2.2 Named entity recognition

Named entity recognition, also known as entity identification, entity chunking or entity extraction, is a subtask of information extraction which focuses on recognizing information units like person names, organizations, locations, products, groups, time and date, percentages, etc. The categorization of a named entity can be different based on the purpose of the task. NER is an important NLP task and is used in many linguistic pipelines as an important part. For example, it is needed as a pre-processing step for NLP tasks like machine translation, question-answering, information retrieval, etc.

The early systems for NER tasks tend to use handcrafted rule-based algorithms, while modern systems usually use supervised and semi-supervised machine learning methods. Supervised learning requires a large amount of annotated data which can be expensive to collect. Semi-supervised and unsupervised learning are proposed to address this problem by using unlabeled data. The dominant approaches nowadays are the combination of CRF and different neural network methods.

2.2.1 Rule-based approaches

Named entity recognition tools that are using rule-based approaches usually build finite state patterns manually. The patterns that are similar to regular expression are aiming to match a sequence of words. Mikheev et al. (1998) utilized eXtensible Markup Language (XML) to simplify the process. Tokenizers that are used here not only divide words by spaces, but also identify tokens according to some agreed definition. For example, “Robert Downey Jr” can be identified as a single token by this tokenizer. Compared to identifying time and number, identifying names is more complicated and depend more on context. To better identify the type of names, tokens will go through five phases:

1. The sure-fire rule makes use of suggestive context to determine if a token is a person or location or organization. For example, “Eagle” will be identified as an organization with suggestive context “shares of Eagle”.

2. Partial match makes sure that all entities found in the first step are also marked if found elsewhere in the text.

3. Rule relaxation applies the symbolic transduction rules. For example, through the first two steps, if a token has been tagged as location, the following capitalized unknown token will also be tagged as location.

4. Another partial match is performed to tag what was missing in step two.

5. The final step is to mark entities in the title of the article.

Humphreys et al. (1998) proposed a rule-based tool LaSIE-II that made use of gazetteer, POS tags, morphological information and semantic tags to build
the parsing grammar rules. The NetOwl built by Krupka and IsoQuest (2005) separated the extraction engine and the extraction configuration so that the extraction engine doesn’t contain any configuration-specific information. The extraction configuration used a generic method to meet the requirement of different types of extraction. It also executed a competition over rules to tackle the ambiguity of names (for example, Washington is a location or a person).

To sum up, rule-based approaches try to handle NER tasks by constructing a sequence of rules based on linguistic knowledge and patterns like regular expressions to extract named entities.

2.2.2 Data-driven approaches

One major problem with the rule-based approaches is that rules are usually made for a specific domain, which makes it hard to be applied elsewhere. As an improvement, data-driven methods are widely adopted nowadays to automatically build rules.

Supervised learning requires a large amount of annotated data which contains both positive and negative examples of named entities. The model studies the features from the annotated corpus to produce the rules automatically to match the given types of named entities. The main approaches used for sequence labeling tasks are supervised learning methods, like Hidden Markov Models (HMM) (Bikel et al., 1998), Maximum entropy Markov models (McCallum et al., 2000), Support Vector Machines (SVM) (Takeuchi and N. Collier, 2002), Decision Trees (Sekine et al., 1998) and Conditional Random Fields (CRF) (McCallum and W. Li, 2003).

Semi-supervised learning uses annotated data as well as unlabeled data. It starts by using some annotated data under supervision to “bootstrapping” the learning process. For example, Brin (1998) first used lexical features to construct regular expressions as the basic rule to start searching book author pairs on web pages. Once the pair has been found, it can often be found with a different format on the same web page. For example, from format “Mark Twain, Huckleberry Finn”, a new format “Huckleberry Finn, by Mark Twain” might be found. Then the system can extract new rules based on the new format to find more pairs. Riloff, Jones, et al. (1999) proposed a mutual bootstrapping method that starts by feeding prepared entity examples of a certain type to the system, then uses the context found around the entities to build new patterns to find new entities.

Unsupervised learning requires only unlabeled data, and the typical approach is clustering. Alfonseca and Manandhar (2002) tried to solve the problem of assigning appropriate NE types in WordNet to unknown entities by first assigning a topic signature to each WordNet synset, then compare the word context of the unknown entity with the topic type to find the most similar one. Shinyama and Sekine (2004) made use of the observation that a named entity usually appears in several news articles in the same period, while common nouns don’t. They compared the words in several newspapers by time-series distributions to find rare named entities, and this method can be used to strengthen other NER methods.
2.2.3 Neural network approaches

Neural network methods are quite popular in NER tasks recently. When using a supervised learning method like CRF and SVM to tackle the NER task, hand-crafted features are needed from experienced linguistic researchers. Using a neural network to extract features automatically makes it possible for laymen to handle NER tasks without too much linguistic knowledge.

Collobert and Weston (2008) proposed a unified convolutional neural network (CNN) architecture which can be trained jointly for multiple NLP tasks including POS tagging, chunking, NER, semantic role labeling, language models and semantically related words (“Synonyms”). Almost all of the tasks use labeled corpora except for language models which leverage a semi-supervised method using unlabeled data. The model mentioned above is built by a lookup table layer, convolution layer, and a softmax layer.

BI-LSTM-CRF architecture has shown great potential in recent years. Z. Huang et al. (2015) introduced the bidirectional LSTM CRF model (BI-LSTM-CRF) to achieve state-of-the-art accuracy on POS, chunking and NER tasks. The model is based on the study of LSTM networks, bidirectional LSTM (BI-LSTM) networks and LSTM with a CRF layer (LSTM-CRF). This work shows that the BI-LSTM-CRF model is efficient because it uses both past and future input features from the bidirectional LSTM layer, and could also use sentence-level tag information from the CRF layer. However, this model still uses some of the hand-crafted features to extract character-level information like spelling features.

As an attempt to build an end-to-end system that includes character-level information, Chiu and Nichols (2016) proposed a BI-LSTM-CNN model which automatically detects character-level and word-level features. Ma and Hovy (2016) introduced a BI-LSTM-CNN-CRF model which leverages the sentence-level tag information from the CRF layer. This model achieves state-of-the-art accuracy on both POS tagging and NER tasks.

All the neural network models mentioned above require a large amount of labeled corpora for feature extraction. Liu et al. (2018) proposed a neural framework to extract information from raw texts without any additional supervision. It combines language models and BI-LSTM-CRF to solve sequence labeling tasks. The model is proved to be efficient. It manages to complete training in 6 hours on a single GPU for CoNLL03 NER task, and also achieves state-of-the-art F1 score of 91.71 +/-0.10.

To sum up, neural network approaches are widely used recently to extract features to feed into the CRF model. LSTM and CNN methods can be used to achieve state-of-the-arts performance, which we are going to experiment with in this thesis.

2.3 Shared tasks for NER on social media text

NER on social media text has received more and more attention from NLP researchers recently. As a result, many tools and methods have been introduced. To evaluate these tools and methods in a standard way, a shared task on NER has been held as part of the Workshop on Noisy User-generated Text (W-NUT) since 2015.
The first shared task in 2015 (Baldwin et al., 2015) used Twitter data as training and development data. The shared task aims to distinguish 10 different named entity types. The training data contains 1,795 annotated tweets and the development data contains 599 annotated tweets. Eight teams participated in this shared task. Almost all of the teams used hand-crafted features. Only one team was using word embeddings and a feed-forward neural-network (FFNN) to generate features (Godin et al., 2015). Most of the teams used the CRF method with word embeddings and Brown clusters as features. There were two teams that chose alternative methods. Cherry et al. (2015) used a semi-Markov tagger, and Yamada et al. (2015) used entity-linking based features. The best result was achieved by Yamada et al. (2015) with F1 score 56.41 using entity-linking based features.

The W-NUT 2016 shared task (Strauss et al., 2016) combined the training and development data used in 2015 shared task as training data and added 1000 more annotated Twitter data as development data. Out of the eight teams that have participated, three teams used CRF with hand-crafted features, four teams used BI-LSTM to automatically extract features, one team used the Learning to search (L2S) method to extract rich features. Limsopatham and N. H. Collier (2016) got the best F1 score 52.41 by using bidirectional LSTM to extract orthographic features.

The W-NUT 2017 shared task (Derczynski et al., 2017) focused on categorizing novel and rare entities into 6 different named entity types. The training data contains 1000 annotated tweets, while development and test data are collected from Reddit, YouTube, and StackExchange. Among the seven teams that have participated, only Williams and Santia (2017) used context models to perform context-sensitive recognition. The rest of the teams used variants of BI-LSTM. The best F1 score 41.86 was achieved by Aguilar et al. (2017) using a multi-task BI-LSTM-CNN network.

To sum up, BI-LSTM-CRF model has been proven to be useful and is becoming more and more popular in NER tasks. The task of recognizing novel and rare entities is still very tough and needs more attention. In this thesis we also experiment with BI-LSTM-CRF model. Unlike the teams that also use this model, we only use features that are automatically extracted from character and word level instead of using additional hand-crafted features.

### 2.4 Neural network model for NER tasks

Neural network methods are widely used within NLP area and LSTM-CRF model is becoming the dominant approach to handle NER tasks. This section will provide a brief introduction of the LSTM-CRF model.

#### 2.4.1 LSTM

The feed-forward neural network was the first and most straightforward type of artificial neural network devised (Schmidhuber, 2015). It contains a single layer of output nodes. The input information moves in only forward, goes through the hidden nodes and then to the output nodes.
For sequence labeling tasks, using a feed-forward neural network is not ideal since it can only handle single data points inputs, while the label of the current token also depends heavily on the context. Recurrent neural networks (RNN) can mitigate this problem by using a loop to persist information. Figure 2.2 shows the unfolding structure of a recurrent neural network, where $x$ represents the input layer, $h$ represents the hidden layer and $o$ represents the output layer. On time $t$, the hidden layer $h_t$ not only receives input vector from $x_t$, it also receives value $V$ from $h_{t-1}$, and provide information to $h_{t+1}$.

Although RNN can make use of context information, it does not perform well in practice since the model gets biased by the nearest input (Bengio et al., 1994). Long short-term Memory (Gers et al., 2002; Hochreiter and Schmidhuber, 1997) is a variant of RNN. Compared to RNN, LSTM uses an input gate, an output gate, and a forget gate to provide more information from long-range dependencies to the memory cell, and forget some of the information. The LSTM memory cell is implemented as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
\[ a_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \]
\[ c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \]
\[ h_t = o_t \odot \tanh(c_t) \]

\( \sigma \) is the element-wise sigmoid function as the gating function with value between 0 and 1, and \( \odot \) is the element-wise product. \( W_i, W_f, W_o, W_c \) contain the input weights connected to each gate, while \( U_i, U_f, U_o, U_c \) contain the recurrent weights matrices corresponding to each gate. The LSTM computes the hidden states \( h_t \) of time \( t \) which contains all useful information from time 0 to \( t \), and all the weights and bias are updated after each iteration. To put it in a simpler way, the state of each gate and the hidden states are calculated based on the input weights and bias that are related to each gate.

2.4.2 Bidirectional LSTM

For sequence labeling task like NER, both past and future context information are important, thus bidirectional LSTM architecture (Graves et al., 2013) is suitable for this task. Figure 2.4 shows a bidirectional LSTM neural network, where \( x \) represents the input layer, \( A \) represents the forward LSTM layer which is used to extract past context features, \( A' \) represents the backward LSTM layer which is used to extract future context features, and \( y \) represents output layer which makes use of both the past and future context features.

2.4.3 Conditional Random Field

CRF is a type of discriminative undirected graph model (Lafferty et al., 2001) that is widely used in sequence labeling tasks. The basic idea of CRF is taking a sequence of feature states as input, and find the most probable label sequence as output. The input feature states are a sequence of features with the same order as the input sentence, and each feature corresponds to each token. Since the CRF model generates the output sequence that maximizes the possibility of the whole sequence instead of each token, the sentence level information is included by using the CRF method.
In our task, suppose we take the output sequence $O = (o_1, o_2, ..., o_n)$ of a BI-LSTM layer as input to a CRF layer, we could expect a predicted label sequence $y = (y_1, y_2, ..., y_n)$ as the final output. Then the conditional probability $p(y|o)$ is defined as follows:

$$p(y|o; W, b) = \frac{\prod_{i=1}^{n} \exp(W_{y_{i-1},y_i}^T o + b_{y_{i-1},y_i})}{\sum_{y' \in \Omega(O)} \prod_{i=1}^{n} \exp(W_{y'_{i-1},y'_i}^T o + b_{y'_{i-1},y'_i})}$$

where $\Omega(O)$ represent the set of all possible label sequences, $W$ and $b$ denote weight and bias respectively.

The maximum conditional likelihood estimation is used to update parameter $W$ and $b$ during the training process as follows:

$$L(W, b) = \sum \log p(y_i|o_i; W, b)$$

Finally, the Viterbi algorithm is used to generate the optimal output sequence $y^*$:

$$y^* = \arg \max_{y \in \Omega(O)} p(y|o)$$
3 Data and experimental setup

3.1 Data and evaluation

3.1.1 Data

The data used in this thesis is from the W-NUT 2017 shared task (Derczynski et al., 2017). The training data used in W-NUT 2017 shared task is from the previous shared task in 2015 (Baldwin et al., 2015) which only contains data from Twitter. The training data includes 3,394 annotated tweets with 62,730 tokens. For development and test data, more data sources are included to cover different domains and also to include longer texts (Twitter has a limitation of 140 characters for one post). The data sources include comments from Reddit, YouTube, and StackExchange. The Twitter data is also included to maintain some connection with the training data. Since the aim of the task is to recognize novel and emerging entities, the entities that are in the training data are excluded from the development and test data to ensure the entities in this two corpora are novel.

Six entity types are defined in this task based on CoNLL (Sang and De Meulder, 2003), ACE (Doddington et al., 2004), MSM (Rizzo et al., 2015):

1. person - Names of persons including first names, surnames, artistic names, etc. The punctuation in the middle of names is also included. Fictional people who have a name can be included.

2. location - Names of locations. The punctuation in the middle of names is also included. Fictional locations which have a name can be included.

3. corporation - Names of corporations. The punctuation in the middle of names is also included.

4. product - Official names of physical products. The punctuation in the middle of names is also included. Fictional products which have a name can be included.

5. corporation - Names of creative works created by a human. The punctuation in the middle of names is also included.

6. group - Names of groups which have a unique name.

The corpus was tokenized using GATE and then processed using max-recall automatic adjudication (Derczynski et al., 2016) before being manually annotated by English native speakers. Statistics for data sets can be seen in Table 3.1.

1https://gate.ac.uk/sale/tao/index.html
3.1.2 Word embeddings
A word embeddings method maps words or phrases to real-valued vectors. In such a way, the vectors capture the semantic and syntactic information of words (Mikolov et al., 2013). There are many implemented word embedding tools. Among those are word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). In this thesis, we use pre-trained word embeddings provided by GloVe\(^2\) since according to Pennington et al. (2014), GloVe performs better than word2vec on named entity recognition tasks. We use two different pre-trained word embeddings based on Common Crawl data, which contains 840 billion tokens and 2.2 million vocabulary and Twitter data which contains 2 billion tweets, 27 billion tokens, and 1.2 million vocabulary.

Out-of-vocabulary words are represented by randomly initialized embeddings from the range \([-\sqrt{\frac{3}{\text{dim}}}, +\sqrt{\frac{3}{\text{dim}}}]\), where dim is the dimension of word embeddings according to He et al. (2015).

3.1.3 Evaluation
According to the task description (Derczynski et al., 2017), the classic precision, recall, and F1 can be used to evaluate the performance of the NER models, and surface forms of the entities as well. That is, for example, the classic way of evaluation would tend to give a high score to a model that successfully recognizes Sweden as a location if most of the location entities are just Sweden. However, when surface forms are taken into consideration of the evaluation, recognising Sweden successfully as a location will only be rewarded once. These two ways of evaluation are denoted as entity and surface, respectively.

3.2 Experimental setup
We experiment on three models that are all based on LSTM-CRF model. The first model we experiment with only uses word embeddings as input to feed into LSTM-CRF model, while the other two models use neural network structures to construct character embeddings and concatenate word embeddings with character

\(^2\)https://nlp.stanford.edu/projects/glove/
embeddings as input. One model uses a BI-LSTM to build character embeddings (Lample et al., 2016), the other uses Convolutional Neural Networks (CNN) (Ma and Hovy, 2016). A more detailed description of the two neural network structures is provided below.

After getting the result of three models, the hyper-parameter optimization is performed on the model that gets the best result.

3.2.1 BI-LSTM for character-level representation

We build the BI-LSTM model to learn character-level features based on the work by Lample et al. (2016) as in Figure 3.1. From the figure we can see that the vector for each character is fed into the forward and backward layers of the BI-LSTM model. The output character-level representation contains the output from both forward layer, which could capture suffix information of the word, and backward layer, which captures prefix information of the word.

3.2.2 CNN for character-level representation

Another model to extract character-level features by using convolutional neural networks (CNN) based on the work by Ma and Hovy (2016) as in Figure 3.2. We can see from the figure that character embeddings for each character are fed into the CNN layer. For words with the length that do not exceed the max length,
padding tokens are used. After going through feature filtering and max pooling, the character-level representation is generated.

**Figure 3.2: Extract character features using CNN**

Source: Ma and Hovy (2016)

3.2.3 Combined character-level and word-level representation

After obtaining the character-level representations by using BI-LSTM or CNN, we concatenate them with the word-level representations which are the classic word embeddings, and then feed the concatenated result as input to the LSTM-CRF model. This architecture is illustrated in Figure 3.1 and 3.3, where we use the word embeddings that contain both character-level and word-level information as the input to BI-LSTM layer, and the CRF layer will output a sequence of labels corresponding to each word.
Figure 3.3: BI-LSTM-CRF model with combined character-level and word-level word embeddings as input

Source: Lample et al. (2016)
4 Experiment results and discussion

4.1 Evaluate between three models

First of all, we evaluate different model architectures with the two pre-trained word embeddings on development data. The results can be found in Table 4.1 and 4.2, where character-level representation extracted by BI-LSTM is denoted as \textit{char-LSTM}, character-level representation extracted by CNN is denoted as \textit{char-CNN}. \textit{cc} and \textit{tt} represent pre-trained word embeddings using Common Crawl and Twitter data respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-CRF + cc</td>
<td>69.51%</td>
<td>34.09%</td>
<td>45.75%</td>
</tr>
<tr>
<td>LSTM-CRF + tt</td>
<td>52.22%</td>
<td>5.62%</td>
<td>10.15%</td>
</tr>
<tr>
<td>char-LSTM-LSTM-CRF + cc</td>
<td>68.34%</td>
<td>45.69%</td>
<td>54.77%</td>
</tr>
<tr>
<td>char-LSTM-LSTM-CRF + tt</td>
<td>47.62%</td>
<td>7.18%</td>
<td>12.47%</td>
</tr>
<tr>
<td>char-CNN-LSTM-CRF + cc</td>
<td>65.66%</td>
<td>38.88%</td>
<td>48.84%</td>
</tr>
<tr>
<td>char-CNN-LSTM-CRF + tt</td>
<td>53.28%</td>
<td>16.51%</td>
<td>25.21%</td>
</tr>
</tbody>
</table>

Table 4.1: Result comparison (entity)

<table>
<thead>
<tr>
<th>Model</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-CRF + cc</td>
<td>67.79%</td>
<td>32.40%</td>
<td>43.84%</td>
</tr>
<tr>
<td>LSTM-CRF + tt</td>
<td>52.27%</td>
<td>6.16%</td>
<td>11.02%</td>
</tr>
<tr>
<td>char-LSTM-LSTM-CRF + cc</td>
<td>67.28%</td>
<td>43.78%</td>
<td>53.04%</td>
</tr>
<tr>
<td>char-LSTM-LSTM-CRF + tt</td>
<td>49.17%</td>
<td>7.90%</td>
<td>13.61%</td>
</tr>
<tr>
<td>char-CNN-LSTM-CRF + cc</td>
<td>64.37%</td>
<td>37.48%</td>
<td>47.38%</td>
</tr>
<tr>
<td>char-CNN-LSTM-CRF + tt</td>
<td>52.44%</td>
<td>17.27%</td>
<td>25.98%</td>
</tr>
</tbody>
</table>

Table 4.2: Result comparison (surface)

From the results we can see that the overall performance of surface evaluation is lower than that of the entity evaluation. This is expected since the same entity with different forms will only be rewarded once with surface evaluation, while different forms of the same entity could be rewarded multiple times with entity evaluation. The score of surface evaluation is lower than that of entity evaluation by a small margin, which is also expected since this task is aiming to recognize previously unseen entities, so there are not too many same entities in the data sets.

In terms of word embeddings, it is shown that GloVe trained by Common Crawl data performs much better than the one trained by Twitter data. Moreover, for the same model, the precision score remains in relatively the same level using the two pre-trained word embeddings, while the recall score dropped substantially.
by using GloVe embeddings trained by Twitter data. This indicates that the model is less sensitive for finding entities by using Twitter data trained word embeddings. This is not expected since the whole task is on social network texts, so Twitter data trained word embeddings are expected to be more informative. We think it is because common crawl corpus contains much larger vocabulary than Twitter data. It could be part of the further work to investigate more into this.

The results show that the LSTM-CRF model gets a lower performance than the other two models. This implies that character-level information plays an important role in NER tasks on social media data. We think this makes sense since on social media people tend to use abbreviations and morphs (Zhang et al., 2015) to represent entities. This kind of information can be better captured by character-level representations than word-level representations.

In comparison with the two models using different neural methods to extract character-level features, we see that the one using BI-LSTM performs better than the CNN model. We think this is because the CNN method is specialized in extracting position-invariant features, and thus suitable for tasks like image recognition (Lample et al., 2016). On the other hand, BI-LSTM can catch prefixes and suffixes by using the forward and backward layer, which is useful in our task since prefixes and suffixes have been proven to be able to help increase the performance for NER tasks (Yadav et al., 2018).

To sum up, the char-LSTM-LSTM-CRF model with word embeddings that are pre-trained by using common crawl corpus outperforms other models with the F1 score of 53.04%. Character-level features play an important role in NER tasks on social media text. BI-LSTM is more suitable than CNN for character-level representation extraction regarding this task.

### 4.2 Hyper-parameter optimization

Since the char-LSTM-LSTM-CRF model gets the best performance among the three models, we will use the development corpus to optimize hyper-parameters on this model to get better results. Table 4.3 shows the hyper-parameters that we experimented with.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Range</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character embedding dimension</td>
<td>[50, 200]</td>
<td>100</td>
</tr>
<tr>
<td>LSTM state size</td>
<td>[50, 250]</td>
<td>100</td>
</tr>
<tr>
<td>Epochs</td>
<td>[15, 40]</td>
<td>25</td>
</tr>
<tr>
<td>Batch size</td>
<td>[10, 50]</td>
<td>20</td>
</tr>
<tr>
<td>Dropout rate</td>
<td>[0.2, 0.7]</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Table 4.3: Hyper-parameter optimization**

Figure 4.1 shows experiment results of optimizing the character embeddings dimension using both entity evaluation and surface evaluation. We explored values between 50 and 200, and we obtain the best F1 score using the value 100.
Figure 4.1: Optimization of the character embeddings dimension

Figure 4.2: Optimization of LSTM state size
The results of the experiments with tuning LSTM state size can be seen in Figure 4.2, where we experimented with value from 50 to 250. The best F1 score was achieved from the value 100.

![Figure 4.2: Optimization of LSTM state size](image)

**Figure 4.2:** Optimization of LSTM state size

Figure 4.3 shows the results of experiments with optimising epochs. We evaluate the results by using values between 15 to 40, where we get the best result from the value of 25. Similarly, in Figure 4.4 you can find the experiments with optimizing the batch size, where the best performance was achieved by setting the value to 20. The range of the value we experimented is from 10 to 50. Finally, we tried values from 0.2 to 0.7 for dropout rate. In Figure 4.5 we see that using value 0.5 we got the best F1 score.

![Figure 4.3: Optimization of epochs](image)

**Figure 4.3:** Optimization of epochs

![Figure 4.4: Optimization of batch size](image)

**Figure 4.4:** Optimization of batch size
4.3 Final test result

After experimenting with different models and hyper-parameter tuning, we finally applied the test corpus to the char-LSTM-LSTM-CRF model with optimal hyper-parameters. We combined training data and development data as training data for the final experiment, and the results that are obtained using entity evaluation and surface evaluation can be found in Table 4.5. Our model achieved F1 score of 42.44% and 40.67% by using the entity evaluation and the surface evaluation respectively. We achieved the best score compared to the other participants in this shared task as shown in Table 4.6. (Derczynski et al., 2017).

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>48.97%</td>
<td>37.44%</td>
<td>42.44%</td>
</tr>
<tr>
<td>Surface</td>
<td>47.23%</td>
<td>35.71%</td>
<td>40.67%</td>
</tr>
</tbody>
</table>

Table 4.4: Final Result

<table>
<thead>
<tr>
<th>Participants</th>
<th>F1(entity)</th>
<th>F1(surface)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIC-CIS</td>
<td>37.06%</td>
<td>34.25%</td>
</tr>
<tr>
<td>Arcada</td>
<td>39.98%</td>
<td>37.77%</td>
</tr>
<tr>
<td>Drexel-CCI</td>
<td>26.30%</td>
<td>25.26%</td>
</tr>
<tr>
<td>SJTU-Adapt</td>
<td>40.42%</td>
<td>37.62%</td>
</tr>
<tr>
<td>FLYTXT</td>
<td>38.35%</td>
<td>36.31%</td>
</tr>
<tr>
<td>SpinningBytes</td>
<td>40.78%</td>
<td>39.33%</td>
</tr>
<tr>
<td>UH-RiTUAL</td>
<td>41.86%</td>
<td>40.24%</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>42.44%</strong></td>
<td><strong>40.67%</strong></td>
</tr>
</tbody>
</table>

Table 4.5: Results comparison
After getting the final experiment results, we did an additional experiment with the same model using only training corpus as training data instead of using both training and development corpus, and we got F1 score 39.97% using the entity evaluation and F1 score 37.67% using the surface evaluation. It indicates that the results could be improved with larger training data sets.

4.4 Error analysis

From Table 4.7 and Table 4.8 we can see the detailed results of the precision, recall and F1 score for each entity types of the final experiment on the test set. The scores using entity evaluation are higher than the scores using surface evaluation in general.

If we take a closer look at the results obtained using entity evaluation, we can see that the model performs best on recognising person names with precision, recall and F1 score of 64.37%, 55.01%, and 59.37% respectively. The precision and recall are both relatively high, which means the model can correctly recognize a large proportion of person names.

*location* gets the lowest precision score of 31.50%, but it achieves the second-highest recall score 53.33%. It indicates that the model tends to falsely recognize entities as locations. Because of the relatively high recall score, *location* also gets the second-highest F1 score 39.60%.

*product* obtains the lowest F1 score 22.22%, while it achieves the second-highest precision score 53.33%. It shows that although the model has the ability to correctly recognize the entity as type *product*, it fails to find enough product names.

The rest of the types *corporation, creative-work, and group* have F1 scores between 24% and 28.5%. They all have higher precision scores compared to their recall scores.

To sum up, the model tends to be precise instead of sensitive to the entities. We could think of ways to make the model recognize more entities in the future.

<table>
<thead>
<tr>
<th>Type</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>corporation</td>
<td>34.29%</td>
<td>18.18%</td>
<td>28.31%</td>
</tr>
<tr>
<td>creative-work</td>
<td>40.26%</td>
<td>21.83%</td>
<td>28.31%</td>
</tr>
<tr>
<td>group</td>
<td>46.55%</td>
<td>16.36%</td>
<td>24.22%</td>
</tr>
<tr>
<td>location</td>
<td>31.50%</td>
<td>53.33%</td>
<td>39.60%</td>
</tr>
<tr>
<td>person</td>
<td>64.37%</td>
<td>55.01%</td>
<td>59.37%</td>
</tr>
<tr>
<td>product</td>
<td>51.43%</td>
<td>14.17%</td>
<td>22.22%</td>
</tr>
</tbody>
</table>

*Table 4.6: Detailed results (entity)*
<table>
<thead>
<tr>
<th>Type</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>corporation</td>
<td>32.14%</td>
<td>15.00%</td>
<td>20.45%</td>
</tr>
<tr>
<td>creative-work</td>
<td>39.73%</td>
<td>21.32%</td>
<td>27.75%</td>
</tr>
<tr>
<td>group</td>
<td>45.61%</td>
<td>18.44%</td>
<td>26.26%</td>
</tr>
<tr>
<td>location</td>
<td>28.51%</td>
<td>52.00%</td>
<td>36.83%</td>
</tr>
<tr>
<td>person</td>
<td>64.26%</td>
<td>52.13%</td>
<td>57.56%</td>
</tr>
<tr>
<td>product</td>
<td>51.61%</td>
<td>13.68%</td>
<td>21.62%</td>
</tr>
</tbody>
</table>

**Table 4.7:** Detailed results (surface)
5 Conclusion and future work

In this thesis, we focus on recognising novel and emerging named entities on social media data, which is in line with the 2017 W-NUT shared task (Derczynski et al., 2017). We decide to use LSTM-CRF model to build an end-to-end system to avoid manually engineering hand-crafted features to handle the noise in the social media data.

After comparing models with and without character embeddings, we can say character-level features are useful for NER task on social media data. We also evaluate two methods of capturing character-level representations: the BI-LSTM method and the CNN method. We find out that the BI-LSTM architecture is more suitable for NER task. By using word embeddings that are pre-trained using different data sets, we conclude that a larger training data set could possibly lead to a better result, which we could explore more in the future.

We further tune the system by optimizing the hyper-parameters on development data, and we achieve the best F1 score of 42.44% on test data among all the participants in this shared task. We also find the model could possibility achieve better performance by increasing the size of the training data. An error analysis of the model has been provided as well.

In the future, we can try to increase the size of the training data to obtain a better performance. Further investigation on the word embeddings which are pre-trained using more Twitter data is also worth a try.
Bibliography


Conover, Michael D, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini (2011). “Political polarization on twitter”. In: Fifth international AAAI conference on weblogs and social media.


Godin, Frédéric, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle (2015). “Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations”. In: Proceedings of the workshop on noisy user-generated text, pp. 146–153.


Liu, Liyuan, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han (2018). “Empower sequence labeling with task-aware neural language model”. In: Thirty-Second AAAI Conference on Artificial Intelligence.


Rizzo, Giuseppe, Amparo Elizabeth Cano Basave, Bianca Pereira, Andrea Varga, Matthew Rowe, Milan Stankovic, and A Dadzie (2015). “Making Sense of Mi-


Yin, Jie, Sarvnaz Karimi, Andrew Lampert, Mark Cameron, Bella Robinson, and Robert Power (2015). “Using social media to enhance emergency situation awareness”. In: Twenty-fourth international joint conference on artificial intelligence.

Zhang, Boliang, Hongzhao Huang, Xiaoman Pan, Sujian Li, Chin-Yew Lin, Heng Ji, Kevin Knight, Zhen Wen, Yizhou Sun, Jiawei Han, et al. (2015). “Context-aware entity morph decoding”. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International