Bidirectional LSTM-CNNs-CRF Models for POS Tagging

Hao Tang
Uppsala University
Hao.Tang.8992@student.uu.se

Uppsala universitet
Department of Linguistics and Philology
Master's Programme in Language Technology
Master's Thesis in Language Technology
October 10, 2018

Supervisors:
Harald Hammarström, Uppsala University
Yan Shao, Uppsala University
Abstract

In order to achieve state-of-the-art performance for part-of-speech (POS) tagging, the traditional systems require a significant amount of hand-crafted features and data pre-processing. In this thesis, we present a discriminative word embedding, character embedding and byte pair encoding (BPE) hybrid neural network architecture to implement a true end-to-end system without feature engineering and data pre-processing. The neural network architecture is a combination of bidirectional LSTM, CNNs, and CRF, which can achieve a state-of-the-art performance for a wide range of sequence labelling tasks. We evaluate our model on Universal Dependencies (UD) dataset for English, Spanish, and German POS tagging. It outperforms other models with 95.1%, 98.15%, and 93.43% accuracy on testing datasets respectively. Moreover, the largest improvements of our model appear on out-of-vocabulary corpora for Spanish and German. According to statistical significance testing, the improvements of English on testing and out-of-vocabulary corpora are not statistically significant. However, the improvements of the other more morphological languages are statistically significant on their corresponding corpora.
Acknowledgements

I would like to thank my supervisors Harald Hammarström and Yan Shao for their continuing support, advice and encouragement. I also would like to thank my family, friends and all my teachers in the Department of Linguistics and Philology at Uppsala University.
## Contents

1 Introduction ......................................................... 5  
  1.1 Background and Purpose ..................................... 5  
  1.2 Thesis Outline .............................................. 6  
2 Neural Network Architecture ................................. 8  
  2.1 CNNs for BPE and Character Representation ............. 8  
  2.2 BI-LSTM-CRF .................................................. 10  
    2.2.1 LSTM .................................................... 10  
    2.2.2 BI-LSTM ................................................ 11  
    2.2.3 CRF ..................................................... 11  
  2.3 BI-LSTM-CNNs-CRF .......................................... 12  
3 Training for the Neural Network .............................. 14  
  3.1 Hyper-Parameters ............................................ 14  
  3.2 Variable Initialisation ..................................... 15  
    3.2.1 Weights and Bias Initialisation ....................... 15  
    3.2.2 Embeddings ........................................... 15  
  3.3 Optimization for Loss ....................................... 16  
  3.4 Dropout Training ........................................... 16  
4 Experiments ......................................................... 17  
  4.1 Datasets ..................................................... 17  
  4.2 Main Results ................................................ 17  
  4.3 Error Analysis ............................................... 19  
  4.4 Related Work ................................................ 22  
5 Conclusion and Further Work .................................. 23
1 Introduction

In the field of natural language processing (NLP), part-of-speech (POS) tagging represents labelling each word in a text with a unique POS tag such as noun, adjective, pronoun, and adverb. The task of POS tagging has been worked on for a few decades because it is crucial for deep natural language understanding and downstream applications. By knowing the POS of a word, it can help us to reveal the likelihood of its neighboring words. For instance, adjectives are proceeded by adverbs and nouns by determiners and adjectives in English. Moreover, POS is used to identify the syntactic structure around a word in a sentence (verbs are generally part of verb phrases) which makes the POS tagging an important part for syntactic parsing (Chen & Manning, 2014; Koo & Collins, 2010; Ma & Zhao, 2012, 2015; McDonald, Crammer, & Pereira, 2005; Nivre & Scholz, 2004) and semantic analysis. In addition, the POS is also important for stemming which is frequently applied in information retrieval. In the NLP summarisation task, the POS is beneficial for selecting specific nouns, adjectives, and verbs from a document. In speech synthesis and recognition, people use POS to produce correct pronunciations. The word record, for instance, is pronounced re/i/cord when it is a verb, re/e/cord when it is a noun. Multi-lingual translation, relation extraction and entity linking (Liu et al., 2017; Luo, Huang, Lin, & Nie, 2015) have also obtained significant progress in part because of the magnificent improvement in the POS tagging.

1.1 Background and Purpose

The main traditional generative models for sequence labelling are Hidden Markov Models (HMM), Maximum entropy Markov models (McCallum, Freitag, & Pereira, 2000) and Conditional Random Fields (CRF) (Lafferty, McCallum, & Pereira, 2001), which achieve relatively good performance (Luo et al., 2015; Passos, Kumar, & McCallum, 2014; Ratinov & Roth, 2009). However, some of these traditional linear statistical models such as CRF and SVM-based models heavily depend on hand-crafted features which are costly to develop. Moreover, these models are not perfectly adoptable to new domains and tasks. Non-linear neural network models with distributed word and character representations have been successful in predicting sequential data in NLP tasks.

Collobert et al. (2011) present a convolutional network model to classify the label of words given contextual information. In recent years, the BI-LSTM-CRF has been successfully applied to sequence labelling tasks in NLP with supervised dataset. Huang, Xu, and Yu (2015) introduced the BI-LSTM-CRF model to achieve state-of-the-art accuracy on POS, chunking and NER. However, they did not consider the character embedding and subword-level embedding by using CNNs in their system. Moreover, they used some hand-crafted features to help their model to improve the accuracy so their model is not a 100% end-to-end system. There are some successful attempts to construct end-to-end systems for English POS, such as BI-LSTM-CNNs (Chiu & Nichols, 2015), BI-LSTM-CNNs-CRF (Ma & Hovy, 2016) and LM-LSTM-CRF framework (Liu et al., 2017). Ma and Hovy (2016) were the first to introduce a novel neural system, BI-LSTM-CNNs-CRF. This model considers both word and character embedding to achieve state-of-the-art accuracy on POS and NER. Liu et al. (2017) proposed a new neural architecture through combining language model and BI-LSTM-CRF for sequence labelling framework. In addition, they used a language model to
leverage character-level knowledge. However, they did not consider the effectiveness of
subword-level information for POS tagging in different languages.

To sum up, the previous work only considers the information extraction on character-
and word level, and they mainly focus on the experiments of neural network models for
English corpora. For the complicated morphological languages which include transfixes,
circumfixes, and duplifixes, the information extraction on character- and word level
would not efficiently capture the correlation among sub-group characters in a word for
identifying its POS. Therefore, the purpose of this thesis is to embed subword-level
representations to the BI-LSTM-CNNs-CRF model. In addition, we try to evaluate how
efficient the subword-level representations can be applied to identify POS for more
morphological languages, especially for their out-of-vocabulary corpora.

1.2 Thesis Outline

This thesis mainly focuses on bidirectional neural network architecture for POS tagging.
It is an end-to-end model without feature engineering or data pre-processing. The input
data for this model consists of three parts: word embeddings, character embeddings, and
embeddings of Byte Pair Encoding (BPE). If pre-trained embeddings are not available
or our memory can not process the large size of embeddings such as the size of German
fastText word embeddings is as large as 5.97G, the model will randomly initialise
the embeddings and update them while training the hyper-parameters. In addition,
Heinzerling and Strube (2017) trained embeddings of Byte Pair Encoding (BPE) for
275 languages by using GloVe\textsuperscript{1} with all Wikipedias\textsuperscript{2}.
Therefore, we directly input the
pre-trained embeddings of BPE to our model.

Word embedding Word embedding is a mathematical high dimensional vector which
is used to represent a corresponding word in the vector space. Each dimension’s
numerical value represents a corresponding feature and may even encode a
semantic or grammatical implication in it (Turian, Ratinov, & Bengio, 2010).
These vector representations can efficiently extract intricate relationships among
words. For example, rooster and hen can be said to be similar in meaning since
they both describe chicken. On the other hand, the two words are regarded as
opposites because they also differ from each other along the primary axis of
sex. These complicated relationships embedded in context has to be represented
by a high dimensional vector which can produce dimensions of meaning; as a
result, it compresses the multi-clustering idea in the distributed representations
(Pennington, Socher, & Manning, 2014).

Character embedding Character embedding indicates a high dimensional vector for
a character which is built from the character n grams among the words that
contain the character. The character n grams are shared across words so character
embeddings incorporate more morphological information on character level,
which can not be included in word embeddings.

Byte Pair Encoding (BPE) The BPE is a compression algorithm that finds the most
frequently occurring pairs of adjacent bytes, and then replaces all instances of the
pair with a byte which was not in the original data (Philip, 1994). As a result, this
\textsuperscript{1}https://nlp.stanford.edu/projects/glove/
\textsuperscript{2}http://attardi.github.io/wikiextractor/
algorithm generates the pair table which contains those pairs of bytes and their corresponding replacement bytes. In addition, BPE is an unsupervised subword segmentation method. Suppose we have a word which is "abcabaacabaa":

The first step: the byte pair "ab" occurs most often so we replace "ab" by a byte which was not in the word, "S". Then we have the following replacement table: ScSaacSaa.

The second step: we replace "aa" by "T" so we have ScSTcST

The third step: we replace "ST" by "Z" so we obtain: ScZcZ

The fourth step: we replace "cZ" by "Q" so we obtain: SQQ

We could stop here.

S = ab
T = aa
Z = abaa
Q = cabaa

From the first step to the fourth step, we can stop at any level which depends on the merge operation of our setting in the BPE algorithm. Given the segmentation setting of the BPE, BPE embedding is a collection of subword embeddings which are pre-trained before feeding into CNNs for training in this thesis. Subword embeddings allow guessing the meaning of unknown and out-of-vocabulary words, and may also indicate the word class of a word. For example, the subword -shire in Merkelfordshire indicates a location, and it is a noun.

Through these three inputs, we want to extract useful information on word-, subword- and character- level to accomplish the task of POS tagging. The baseline model of this thesis is a BI-LSTM-CNNs-CRF. First of all, we need to apply convolutional neural networks (CNNs) (Le Cun et al., 1989) to encode character- and subword- level information of a word into its corresponding representations. The inputs of the CNNs are character embedding and BPE embedding. Through the CNNs, we obtain two word representations which contain character- and subword- level information respectively. Then we concatenate the pre-trained word representation and the previous two word representations processed by CNNs before feeding them as inputs of BI-LSTM-CRF. Finally, we use CRF and and the Viterbi algorithm to assign POS labels optimally over the entire sentence.

The contributions of this thesis: first of all, this neural network architectures help us to extract useful information from character-, subword-, and word levels for POS tagging task in NLP, and its extension helps us to improve the accuracy of POS tagging task on more morphological languages such as Spanish and German than English; Secondly, through testing the corpora of different languages, we found that our model obtains statistical significant improvement on more morphological languages which is not observed on English. Thirdly, we are the first to introduce BPE sub-word embeddings to BI-LSTM-CNNs-CRF model.
2 Neural Network Architecture

In this section, the architecture of the CNNs and BI-LSTM-CRF will be introduced.

2.1 CNNs for BPE and Character Representation

A CNN can be used to effectively extract morphological information such as suffix and prefix based on the order of characters in a word (Chiu & Nichols, 2015; Santos & Zadrozny, 2014). As an extension, we not only use a CNN to extract character-level representation of a given word, but also subword-level representation. Compared with CNNs of character representation, CNNs of BPE not only can identify the relation between POS and suffix and prefix, but also can capture any transfixes, circumfixes, and duplifixes. Moreover, it also allows the system to guess the word class of unknown or out-of-vocabulary words.

Figure 1 and 2 shows the procedure of extracting character-level information based on BPE embeddings and character embeddings. In Figure 1, a given word \( w \) consists of \( L \) characters \( \{c_1, c_2, \ldots, c_L\} \), and each character \( c_l \) is represented by a character embedding \( e_l \) of \( d^{\text{chr}} \) dimensions. The length of the inputs of a sample is set to a fixed number called maximum length, \( m \), the uniform length of each input sample. The maximum length is equal to the longest word of the corresponding dataset. For each input of a sample (a character or a bit of BPE), it is an embedding, \( e_l \) of \( d^{\text{chr}} \) dimensions. For the word that is not as long as the maximum length, we use padding tokens to fill the indices that are outside of the word boundaries. For instance, given the maximum length is 20, the word ‘Brave’ can be separated into 5 characters so the remaining 15 positions will be filled by padding tokens. Character embeddings are encoded by column vectors in the character embedding matrix \( M_w \in \mathbb{R}^{d^{\text{chr}} \times m} \), which is the ‘Character Embedding’ in Figure 1. The CNN applies filtering features and max pooling by a matrix-vector operation to each window of size \( r^{\text{chr}} \), and the window will go through the sequence \( \{e_1, e_2, \ldots, e_m\} \). Define the \( z_h \in \mathbb{R}^{d^{\text{chr}} \times r^{\text{chr}}} \) as the concatenated matrix including the character embedding \( e_h \), where \( h \) indicates the \( h \)th character of the input word, its \((r^{\text{chr}}-1)2\) left neighbors, and its \((r^{\text{chr}}-1)2\) right neighbours:

\[
z_h = (e_{h-(r^{\text{chr}}-1)2}, e_{h+(r^{\text{chr}}-1)2})
\]

Then going through the process of Filtering and Max Pooling, the CNN generates a new character-level embedding of the word which contains local features around each character of the word (Santos & Zadrozny, 2014). Figure 2 shows that the same procedure will be implemented on subword-level by using BPE embeddings as inputs.

Finally, we concatenate the two word representations processed by CNNs together with their corresponding pre-trained word representation. As a result, the concatenated word representation contains character-level, subword-level, and word-level information, and it is prepared as the input of BI-LSTM unit. The CNNs are the same as the one in (Chiu & Nichols, 2015), except that we implement BPE embeddings and character embeddings as the inputs of CNNs.
Figure 1: Convolutional Neural Networks for morphological feature extraction through inputing character embeddings.

Figure 2: Convolutional Neural Networks for morphological feature extraction through inputing BPE embeddings.
2.2 BI-LSTM-CRF

2.2.1 LSTM

In this thesis, we use Long short-term Memory (Graves & Schmidhuber, 2005; Hochreiter & Schmidhuber, 1997) to implement the task of POS tagging. LSTM is a variant of recurrent neural networks (RNN). Compared to an RNN, a LSTM unit is equipped with input gate, output gate, and forget gate facilitating the detection of long-range dependencies remembering the correlations over arbitrary time intervals. In a bidirectional neural network, the gates in each LSTM unit are thought as regulators of the flow of values that goes through the chain of LSTM units. The LSTM memory cell updating its states at time t is calculated as the following:

\[ i_t = \sigma(W_i h_{t-1} + G_i x_t + b_i) \]

\[ f_t = \sigma(W_f h_{t-1} + G_f x_t + b_f) \]

\[ c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c h_{t-1} + G_c x_t + b_c) \]

\[ o_t = \sigma(W_o h_{t-1} + G_o x_t + b_o) \]

\[ h_t = o_t \odot \tanh(c_t) \]

where \( \sigma \) is the element-wise sigmoid function, used as the gating function of the 3 gates (in, out, forget) in LSTM with value between 0 and 1, and \( \odot \) is the pair-wise product for two matrices. \( x_t \) represents the input vector of time t, and \( h_t \) represents the hidden states of time t restoring all the useful information from time 0 to t. \( G_i, G_f, G_c, \) and \( G_o \) represent the weight matrices corresponding to different gates in the LSTM unit controlling information flow. Moreover, \( W_i, W_f, W_c, \) and \( W_o \) are the weight matrices for current hidden layer \( h_t \) and previous hidden layer \( h_{t-1} \). At last, \( b_i, b_f, b_c, \) and \( b_o \) represent the biases. \( W, G, \) and \( b \) are needed to be trained and updated for each batch input of the model.

![LSTM Unit Diagram](image)

**Figure 3:** LSTM Unit
2.2.2 BI-LSTM

For the task of POS, considering both past (Forward LSTM layer in Figure 4) and future (Backward LSTM layer in Figure 4) contexts will be beneficial to the performance. Given a current input feature, its POS tagging is not only influenced by its previous inputs, but also the following inputs after it. This bidirectional LSTM can efficiently restore the past and future information to identify current state (Dyer, Ballesteros, Ling, Matthews, & Smith, 2015). This BI-LSTM can be trained by back-propagation through time (BPTT) (Boden, 2002). In each BI-LSTM unit, there are two separate hidden states remembering the past and future information given current input. In each batch of training, the hidden states will be updated. Finally, the two hidden states are concatenated as one output after the training. In our model, the input for BI-LSTM is a concatenated embedding including word, subword, and character representations in which the subword and character representations are processed through previous CNNs. Suppose the pre-trained word embedding is \( W_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{il}) \), the subword embedding processed by CNNs is \( S_i = (s_{i1}, s_{i2}, \ldots, s_{in}) \), and the character embedding processed by CNNs is \( C_i = (c_{i1}, c_{i2}, \ldots, c_{im}) \) where \( l, n, \) and \( m \) represent the dimension of corresponding embedding and \( i \) represents the \( ith \) word in a sentence. As a result, the concatenated word representation is \( T_i = (W_i, S_i, C_i) \), which is the input of BI-LSTM. In this model, we set the length of each sentence to a fixed value, which is called maximum time step. If the maximum time steps is set to 50, it means there are 50 units of BI-LSTM sequentially connected with each other. When the length of a sentence is less than 50, the indices will be filled by padding tokens outside of the sentence boundaries. Otherwise, we only process the first 50 words of the sentence. Each BI-LSTM has two separate hidden layers: forwards and backwards which are used to capture past and future information individually (See Figure 4). The two hidden layers are concatenated to form the final output. In this model, we set the hidden states = 200 for each LSTM. Because we have two hidden layers: Forward and Backward, the size of the output for each LSTM is \( 2 \times 200 = 400 \) after concatenating. Suppose the forward and backward hidden states of each BI-LSTM are \( F_i = (a_{i1}, a_{i2}, \ldots, a_{ik}) \) and \( B_i = (b_{i1}, b_{i2}, \ldots, b_{ik}) \) where \( k \) represents the state size of each BI-LSTM unit and \( i \) represents the \( ith \) BI-LSTM unit. The final output of two concatenated hidden layers for the \( ith \) BI-LSTM unit is \((F_i, B_i)\).

2.2.3 CRF

The CRF (Lafferty et al., 2001) is a discriminative model that can capture a large amount of interacting and dependent features by taking context into account. By using CRF, we try to optimise the output of sequence labelling over sentence level instead of individual positions. Given a sentence, the optimal sequence of labels are determined by neighbourhoods and grammatical structure of each sentence. As a result, adding CRF can generate higher accuracy rate in general compared with BI-LSTM-CNN model(Huang et al., 2015; Ma & Hovy, 2016).

Suppose the compound output from word level BI-LSTM is \( O_i = (o_{i1,1}, o_{i1,2}, o_{i1,3}, \ldots, o_{i,n}) \) where \( n \) represents the LSTM state size and \( i \) represents the \( ith \) word in a sentence. CRF method states the probability of a sequence POS tags given the BI-LSTM output over a sentence. This is denoted by \( p(\hat{y} | O) \) where \( \hat{y} \) represents the predicted sequence of POS tags and \( O \) is the BI-LSTM output.
Similar to Ma and Hovy (2016), the probability is defined as follows.

\[
p(\hat{y} | o_i) = \frac{\prod_{j=1}^{n} \varphi(\hat{y}_{j-1}, \hat{y}_j, o_j)}{\sum_{y' \in \Omega(O)} \prod_{j=1}^{n} \varphi(y'_{j-1}, y'_j, o_j)}
\]

(2.1)

\(\Omega(O)\) denotes the set of all possible combinations of label sequences,

\[
\varphi(\hat{y}_{j-1}, \hat{y}_j, o_j) = \exp(W_{y_{j-1}, y_j} o_j + b_{y_{j-1}, y_j})
\]

where \(W_{y_{j-1}, y_j}\) and \(b_{y_{j-1}, y_j}\) represent the weight and bias parameters.

We want to minimize the negative log-likelihood in the training process, and update the parameters, \(W\) and \(b\).

\[
\zeta_{CRF} = -\sum \log p(y_i | O_i)
\]

(2.2)

At last, after training the parameters, \(W\) and \(b\), we use this model to find out the most likely sequence \(y^*\) maximising the likelihood.

\[
y^* = \arg \max_{y \in \Omega(O)} p(y | O)
\]

(2.3)

The Equation 2 and 3 can be calculated through the Viterbi algorithm.

2.3 BI-LSTM-CNNs-CRF

At last, we need to connect the concatenated hidden states of LSTM to CRF layer. Figure 4 shows the structure of our neural network model. First of all, for each input word, we obtain its character- and subword level representations through CNNs in Figure 1 and 2. The inputs of CNNs are character and subword embeddings padding as the same length for each word. After the process of CNNs, the character- and subword-level representation vectors are concatenated with the pre-trained word embedding vector. As a whole, it is fed into BI-LSTM network. Finally, we input the concatenated hidden states of BI-LSTM to the CRF layer to jointly decode the optimal POS tagging for an entire sentence.
Figure 4: The main structure of BI-LSTM-CNNs-CRF model. The BPE and character embedding for each word are processed by CNNs extracting the character-level features in Figure 1 and 2. Then we concatenate the BPE, character, and word representations together in order to feed it into the BI-LSTM cells.
3 Training for the Neural Network

In this section, we provide details about the hyper-parameters and initial values setting for this model. This model is implemented by the TensorFlow 1.5.0 library (Abadi et al., 2016). It is trained 30 epochs for each experiment.

3.1 Hyper-Parameters

<table>
<thead>
<tr>
<th>Hyper-Parameters</th>
<th>English</th>
<th>Spanish</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character embedding dimension</td>
<td>300</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Word embedding dimension</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>BPE embedding dimension</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Window size</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LSTM state size</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Conv. filter number</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Optimiser</td>
<td>Adagrad</td>
<td>Adagrad</td>
<td>Adagrad</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Decay rate</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Dropout rate</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Batch size</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Maximum time step</td>
<td>50</td>
<td>150</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 1: Hyper-Parameters

Table 1 shows the hyper-parameters setting in our model. The combination of the hyper-parameters are learned and turned from previous successful experiments. Santos and Zadrozny (2014) introduced the method of extracting character- and subword- level information through CNNs, and the CNN’s hyper-parameters was embedded in the model. LSTM state size, Optimiser, Initial learning rate, Decay rate, Dropout rate, and Batch size were tuned by random search across the space of full hyper-parameters (Ma & Hovy, 2016), so we assigned the same values as theirs. However, we set character embedding dimension as 100 and Conv. filter number as 50 instead of their choices, 30 and 30 respectively. As a result, we try to use 100 dimensional vector and 50 filters to capture more morphological features in CNNs. The Maximum time step refers to the fixed input length for each sentence in our model. Suppose a character embedding of $d_{chr}$ dimensions and window size $w$ so the size of convolutional filter in the CNNs, $f$, is $f \in \mathbb{R}^{(2w+1) \times d_{chr}}$. The dimensional size of randomly initialised embeddings for word and character are uniformly set to 100. Moreover, after exploring the corpora of English, Spanish, and German, I found that for the Spanish datasets there are significant amount of sentences contains more than 50 words so I set the Maximum time step to 150 for Spanish which are different from English’s and German’s setting, 50.
3.2 Variable Initialisation

3.2.1 Weights and Bias Initialisation

Weight matrices are randomly assigned with uniform distribution over \([-x, x]\), where \(x = \sqrt{\frac{6}{r + c}}\) and \(r\) and \(c\) represent the number of rows and columns (Glorot and Bengio, 2010). Bias is set to zero, except the bias for CNNs, which is set to 0.1.

3.2.2 Embeddings

Pre-trained Embeddings:

Word Embeddings: we use GloVe 100-dimensional English embeddings as the word representations, which is trained on 6 billion words from Wikipedia and web text. Given Bi-LSTM-CNNs-CRF setting, the GloVe 100 dimensional embeddings outperformed other public embeddings: Senna, Word2Vec, and the randomly sampled one (Ma & Hovy, 2016). Therefore, in the experiment section, we only test our model with GloVe 100-dimensional word embeddings. For Spanish, we use fastText pre-trained 100-dimensional word representations as input, which is trained on Common Crawl and Wikipedia.3

Character Embeddings: the pre-trained character embeddings for English are partial of ASCII characters, and the training method and dataset is the same as the GloVe pre-trained word embeddings. There are 108 characters (including both lower case and upper case), and each character is represented by a vector in dimension 300.4 There is one parameter needed to be tuned which is the window size. The variable window size determines how much contextual information we need to take into account for a pivot character. In Figure 1, it shows that if we set window size = 2, and pivot character is B, the pivot representation will consider the information of the previous two padding representations and the following representations of \(r\) and \(a\) through CNNs. As a result, after training, each character representations not only contains the specific features of itself, but also its neighbourhoods’. In general, the character embeddings are trained with the hyper-parameters of the neural network (Liu et al., 2017; Ma & Hovy, 2016; ?). In this model, we choose to use pre-trained character embeddings as the initial input of CNN for English, but we randomly initialise the Spanish and German character 100-dimensional embeddings, and then they are updated while training the hyper-parameters of the neural network.

BPE embeddings: BPE is an unsupervised sub-word segmentation algorithm, and it is a useful method to analyse a connection between the word class and the most frequently co-occurring characters (Philip, 1994; Sennrich, Haddow, & Birch, 2016). We use a Python library, sentencepiece5, to classify the words into parts. There are 8 merge operations to decide how sparsely we want to classify a word into. The range of merge operations is from 1000 to 2000006. For the lower merge operation, the sentencepiece library divides a word into more pieces and vice versa. For instance, setting merge operation to 1000 separates "railway" into three parts: r, ail, and way. On the other hand, changing the merge operation to 200000, the separation becomes one part: "railway".

---

3 https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md
4 http://minimaxir.com/2017/04/char-embeddings/
5 https://github.com/google/sentencepiece/tree/master/data
6 https://github.com/bheinzerling/bpemb
Therefore, we need to find out the optimal combination of the merge operation and window size as a whole to deliver the best performance. Moreover, the byte pairs are presented by real numbers in the Python library, sentencepiece; therefore, we need to incorporate it with another Python library, KeyedVectors\(^7\), to extract the pre-trained 100-dimensional BPE embeddings for the corresponding byte pairs before feeding them into CNNs.

**Randomly Initialised Embeddings:**

Word Embeddings: For German, we cannot use pre-trained word embeddings because the size of embeddings is as large as 5.97G, which cannot be processed by our limited memory, 32G, exceeding the limit of our memory before training. Therefore, we randomly initialise all the German word embeddings as 100-dimensional vectors where embeddings are uniformly sampled over \([-x, x]\), where \(x = \sqrt{\frac{3}{\text{dim}}}\) where dim is the dimension of embeddings (Ma & Hovy, 2016).

Character Embeddings: German and Spanish character embeddings are initialised over uniform distribution \([-x, x]\), where \(x = \sqrt{\frac{3}{\text{dim}}}\) where we set \(\text{dim} = 100\).

<table>
<thead>
<tr>
<th>Embeddings</th>
<th>English</th>
<th>Spanish</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Pre-trained GloVe</td>
<td>Pre-trained fastText</td>
<td>Randomly Initialised</td>
</tr>
<tr>
<td>Character</td>
<td>Pre-trained 300-dim</td>
<td>Randomly Initialised</td>
<td>Randomly Initialised</td>
</tr>
<tr>
<td>BPE</td>
<td>Pre-trained 100-dim</td>
<td>Pre-trained 100-dim</td>
<td>Pre-trained 100-dim</td>
</tr>
</tbody>
</table>

*Table 2: The inputs of the neural network models.*

Table 2 indicates the inputs of the neural network models for different languages in this thesis. For English, we choose pre-trained 100-dimensional GloVe word embeddings and 300-dimensional GloVe character embeddings as the inputs. For Spanish, we choose pre-trained 100-dimensional fastText word embeddings and randomly initialised 100-dimensional character embeddings as the inputs. For German, we choose to feed word and character embeddings with randomly initialised 100-dimensional embeddings for the neural network models. As shown in Table 2, all BPE embeddings are pre-trained 100-dimensional embeddings (Heinzerling & Strube, 2017). Because we only compare the different neural network models' performances with the same inputs for a specific language, the variation of the inputs for different languages will not influence our analysis and comparison.

### 3.3 Optimization for Loss

Adagrad (Duchi, Hazan, & Singer, 2011) is used to minimize the loss with initial learning rate 0.05 updating with decay rate \(\varphi_t=0.05\) as training epoch \(t\) increases, \(\eta_t = \frac{\eta_0}{\eta_t + \varphi_t}\) where \(\eta_0 = 0.05\) is initial learning rate.

### 3.4 Dropout Training

In order to avoid overfitting, we implement dropout regularisation to the output of BI-LSTM layer with a fix value of 0.5 (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

\(^7\)https://github.com/bheinzerling/bpemb
4 Experiments

4.1 Datasets

For the POS tagging task, we use Universal Dependencies (UD) datasets of English, Spanish, and German, which includes 17, 18, 16 POS tags according to its kind. The dataset for each language is classified into three parts: training, development, and test. In Table 3, 4, and 5, it shows the details of the UD corpus. The column OOTV shows the number of out-of-the-training-vocabulary words (OOTV) in development dataset or test dataset. The last column OOBV shows the number of out-of-the-training and dev-vocabulary (OOBV) words in testing dataset.

4.2 Main Results

First of all, we test the possible combinations of window size and BPE merge operations for our model on English given other hyper-parameters unchanged. Table 6 shows that setting window size of 2 and BPE of 10000 obtains better performance than other settings. As our previous explanation, when the window size is small, it implies we consider less contextual information at character level. On the other hand, if we set BPE at a small merge operation, it sparsely separates a word into pieces providing the same function as character embedding. In order to extract all the morphological information efficiently, our best setting should have small window size and medium BPE because if we set BPE at a large value, its function will be similar to word embedding. As a
Table 6: BI-LSTM-CNNs-CRF-BPE on the Dev dataset for English.

<table>
<thead>
<tr>
<th>Window Size</th>
<th>BPE: 3000</th>
<th>BPE: 5000</th>
<th>BPE: 10000</th>
<th>BPE: 25000</th>
<th>BPE: 50000</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC.</td>
<td>ACC.</td>
<td>ACC.</td>
<td>ACC.</td>
<td>ACC.</td>
<td>ACC.</td>
</tr>
<tr>
<td>1</td>
<td>0.949893</td>
<td>0.949314</td>
<td>0.950843</td>
<td>0.950265</td>
<td>0.950058</td>
</tr>
<tr>
<td>2</td>
<td>0.950595</td>
<td>0.949479</td>
<td>0.951050</td>
<td>0.949934</td>
<td>0.949851</td>
</tr>
<tr>
<td>3</td>
<td>0.949272</td>
<td>0.948735</td>
<td>0.948073</td>
<td>0.950430</td>
<td>0.949148</td>
</tr>
<tr>
<td>4</td>
<td>0.949148</td>
<td>0.949396</td>
<td>0.950430</td>
<td>0.948611</td>
<td>0.948280</td>
</tr>
<tr>
<td>5</td>
<td>0.949810</td>
<td>0.949934</td>
<td>0.949396</td>
<td>0.950099</td>
<td>0.947908</td>
</tr>
</tbody>
</table>

Table 7: Results of different neural network models with identical inputs.

<table>
<thead>
<tr>
<th></th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI-LSTM (without character embedding)</td>
<td>0.9118</td>
<td>0.9138</td>
</tr>
<tr>
<td>BI-LSTM-CNNs</td>
<td>0.9433</td>
<td>0.9473</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.9471</td>
<td>0.9501</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.9485</td>
<td>0.9513</td>
</tr>
</tbody>
</table>

whole, we try to extract the useful information from different perspectives in our model: character-level, subword-level, and word-level. At last, the output interface, CRF, will consider all the information at sentence-level to optimise the sequence labelling.

We compare our model with other systems: BI-LSTM, BI-LSTM-CNNs, and BI-LSTM-CNNs-CRF. Table 7 shows that BI-LSTM’s performance is lower than other models’ by at least 3.2 percentage points on both dev and test datasets. This implies that the character embeddings plays an important role for POS tagging. Moreover, the CRF can improve the accuracy by 0.6 percentage points and 0.3 percentage points on dev and test datasets respectively. Our model, BI-LSTM-CNNs-CRF-BPE, delivers the best performance, 95.1050 percentage points and 95.0664 percentage points respectively. However, the differences of performance between BI-LSTM-CNNs-CRF-BPE and BI-LSTM-CNNs-CRF is not obvious. After implementing null hypothesis significance testing we find out that for the testing dataset the p-value is 0.7538 so this result would be deemed not statistically significant and, the hypothesis that the model with BPE has the same performance as the one without BPE’s would not be rejected given a significance level of 0.05. Moreover, for the development dataset, the p-value is 0.5549 so it is not statistically significant to reject the hypothesis as well. To sum up, BI-LSTM-CNNs models significantly outperform the BI-LSTM model. By adding CRF and BPE, our model obtains improvements over BI-LSTM-CNNs so jointly decoding and extracting information at subword-level can largely benefit the accuracy of POS tagging for English. However, the improvement of BI-LSTM-CNNs-CRF-BPE over BI-LSTM-CNNs-CRF is not statistically significant for development and test datasets.

For English, we obtain a slight improvement by adding BPE on BI-LSTM-CNNs-CRF architecture, which is not statistically significant so we try to experiment two more morphological languages: Spanish and German. For Spanish, the overall performance on development and test datasets are much higher than English’s. Table 8 shows that the gap between the two different architectures becomes larger than English’s as well.

8https://en.wikipedia.org/wiki/Student%27s_t-test#Independent_two-sample_t-test
which are 0.49 percentage points and 0.96 percentage points. For the testing dataset, the p-value of Spanish is 3.847e-08 so this result would be deemed statistically significant and, the hypothesis that the model with BPE has the same performance as the one without BPE’s would be rejected given a significance level of 0.05. Moreover, for the development dataset, the p-value of Spanish is 3.9025e-07; therefore, it is statistically significant to reject the hypothesis as well.

<table>
<thead>
<tr>
<th></th>
<th>Dev ACC.</th>
<th>Test ACC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.9775</td>
<td>0.9755</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.9824</td>
<td>0.9815</td>
</tr>
</tbody>
</table>

Table 8: Performance on Spanish POS.

For the experiments of German, because the size of pre-trained German word representations is as large as 5.97G, which can not be processed by our limited memory, 32G. It will exceed the limit of our memory before training. Therefore, we randomly initialise all the word embeddings with dimensional size 100. The word embeddings are updated as we train the hyper-parameters. Table 9 shows that the overall performances on development and test datasets for German is lower than the English’s. However, the gap of performances between the two architectures is larger than those of English and Spanish. In Figure 8, it shows that the gap of the two architectures for development and test datasets are 0.67 and 0.69 percentage points respectively. For the testing dataset, the p-value is 0.0009680 so this result would be deemed statistically significant, and the hypothesis that the model with BPE has the same performance as the one without BPE’s would be rejected given a significance level of 0.05. However, for the development dataset, the p-value is 0.0813 so it is not statistically significant to reject the hypothesis.

<table>
<thead>
<tr>
<th></th>
<th>Dev ACC.</th>
<th>Test ACC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.9332</td>
<td>0.9274</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.9389</td>
<td>0.9343</td>
</tr>
</tbody>
</table>

Table 9: Performance on German POS.

4.3 Error Analysis

In the English test dataset, the number of out-of-the-training-vocabulary (OOTV) words is 5186, and the number of out-of-the-both training and development-vocabulary (OOBV) words is 2061. Table 10 shows that the performance of the model with BPE is slightly better than the model without BPE on different subsets of words. For the OOTV dataset, the p-value is 0.3729 so the result would be deemed not statistically significant, and the hypothesis that the model with BPE has the same performance as the one without BPE’s would be not rejected given a significance level of 0.05. For the OOBV dataset, the p-value is 0.1030 so it is not statistically significant to reject the
hypothesis. To sum up, the improvements on different out-of-vocabulary subsets are not statistically significant for English POS tagging.

<table>
<thead>
<tr>
<th></th>
<th>Testing Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OOTV</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.8119</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.8133</td>
</tr>
<tr>
<td></td>
<td>OOBV</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.8049</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.8074</td>
</tr>
</tbody>
</table>

**Table 10:** Comparison of performance on different out-of-vocabulary subsets of words (accuracy for POS).

Figure 5 and 6 show that the BPE model is good at tagging the PROPN and VERB on the corresponding English subsets, and the model without BPE has better performance on ADJ and NOUN. For instance, the performance of the model with BPE achieved 643 correct tags of the PROPN on OOTV dataset; on the other hand, the one without BPE only obtained 618 correct tags on PROPN. Meanwhile, the model without BPE obtained 142 and 586 correct tags on the ADJ and NOUN respectively which are more than the BPE model’s 134 and 560. According to the confusion matrices, we can conclude that both the model with and without BPE have some space to be improved for tagging PROPN and NOUN because most of errors occurred on these two tags. In addition, there is a significant number of ADJ was incorrectly tagged as NOUN, PROPN, and VERB. For example, in Figure 7, it shows that there are 28, 8, and 7 ADJ were incorrectly tagged as NOUN, PROPN, and VERB respectively in the model with BPE; meanwhile, there are 20, 6, and 7 ADJ were incorrectly tagged in the one without BPE. Based on the error analysis, we found that the model would obtain better performance if we can distinguish the PROPN and NOUN more efficiently.

Table 11 shows that the model with BPE outperforms the one without BPE on both OOTV and OOBV for Spanish testing dataset. The model with BPE improves the accuracy by 1.7 for OOTV and 1.62 for OOBV. The confusion matrix in Figure 7 shows the details of performances of Spanish POS tagging on OOTV. In general, PROPN, NOUN, VERB, and ADJ account for the majority of the out-of-vocabulary words for Spanish datasets. When we look at the diagonals of the two tables, the number of correct label on corresponding tags, the model with BPE outperforms the one without BPE on all kinds of tags. According to both of the confusion matrices, we can conclude that both of the models have the worst performances on distinguishing NOUN from ADJ, PROPN, and VERB. For the model with BPE, it mislabeled 28, 24, and 10 NOUN as ADJ, PROPN, and VERB respectively. Meanwhile, the model without BPE obtained slightly worse performance: mislabelling 35, 30, and 8 NOUN as ADJ, PROPN, and VERB respectively. Moreover, the two models mislabeled 63 and 62 ADJ as NOUN which are the worst performance in frequency and proportion of errors. In Figure 8, it shows the confusion matrices of OOBV for Spanish testing dataset, and they have the same patterns as OOTV’s. Therefore, the model with BPE obtains better performances on Spanish out-of-vocabulary datasets. The two architectures mislabeled some NOUN as ADJ, PROPN, and VERB, but the biggest challenge of the architectures on Spanish datasets is distinguishing ADJ and NOUN. For the OOTV dataset, the p-value is 6.977e-08 so this result would be deemed statistically significant, and the hypothesis that the model with BPE has the same performance as the one without BPE’s would be rejected.
given a significance level of 0.05. For the OOBV dataset, the p-value is 1.4371e-07 so it is statistically significant to reject the hypothesis.

<table>
<thead>
<tr>
<th>Testing Dataset</th>
<th>OOTV</th>
<th>OOBV</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.9101</td>
<td>0.9117</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.9271</td>
<td>0.9279</td>
</tr>
</tbody>
</table>

Table 11: Comparison of performance on different out-of-vocabulary subsets of words (accuracy for Spanish POS).

Table 12 indicates that the model with BPE obtains higher accuracy for German on different out-of-vocabulary datasets. The differences between the two architectures become larger than English’s and Spanish’s. For German, the model with BPE increased accuracy by 2.24 and 2.45 percentage points for OOTV and OOBV datasets respectively. First of all, the diagonals of the confusion matrices in Figure 9 and 10 show that the model with BPE is beneficial to correctly label all kinds of tags. Secondly, according to the confusion matrices, we observe that most of the errors occurred between NOUN and PROPN for German. Figure 9 shows that there are 36 PROPN were mislabeled as NOUN, and there are 61 NOUN were mislabeled as PROPN. However, there is above half of ADV was mislabeled in proportion of errors, 39 out of 71 and 43 out of 71 in confusion matrices of Figure 9. Therefore, we can conclude that most of the out-of-vocabulary words consists of NOUN, PROPN, ADJ, and VERB for German. NOUN and PROPN have the highest frequency of mislabelling, but ADV has the relatively high mislabelling rate in proportion of errors. For the OOTV dataset, the p-value is 2.4152e-17 so this result would be deemed statistically significant, and the hypothesis that the model with BPE has the same performance as the one without BPE’s would be rejected given a significance level of 0.05. For the OOBV dataset, the p-value is 1.2849e-19 so it is statistically significant to reject the hypothesis.

<table>
<thead>
<tr>
<th>Testing Dataset</th>
<th>OOTV</th>
<th>OOBV</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI-LSTM-CNNs-CRF</td>
<td>0.8486</td>
<td>0.8480</td>
</tr>
<tr>
<td>BI-LSTM-CNNs-CRF-BPE</td>
<td>0.8710</td>
<td>0.8725</td>
</tr>
</tbody>
</table>

Table 12: Comparison of performance on different out-of-vocabulary subsets of words (accuracy for German POS).

To sum up, the improvement of our model on different out-of-vocabulary datasets of English is not significant. However, the improvements obtain statistical significance for German and Spanish. Secondly, based on the error analysis, we find that most of mislabelling occurs between PROPN and NOUN in frequency of errors. But when we analyse the errors in proportion, the distribution of mislabelling has different patterns on different languages. For English, the proportional mislabelling between PROPN and NOUN is the highest. However, for Spanish, the highest proportional mislabelling
occurs between NOUN and ADJ. For German, the highest proportional mislabelling is between ADJ and ADV.

4.4 Related Work

Our system is built upon the achievement of LSTM-CNNs-CRF model and is further improved to better capture the relationship between POS tagging and subword structures through BPE embedding. The best accuracy of BI-LSTM-CNNs-CRF (Ma & Hovy, 2016) is as high as 97.55% for POS tagging, and the best accuracy of LM-LSTM-CRF framework (Liu et al., 2017) achieves 97.59% for POS tagging. One important reason that based on the same neural network architectures their accuracy is higher than ours is their models implement larger training, development, and testing datasets. Their POS datasets are from the Wall Street Journal (WSJ) portion of Penn Treebank (PTB) containing 38219 sentences for training, 5527 sentences for development, and 5462 sentences for test. However, we implement UD datasets to experiment our model which contains 12250 sentences for training, 1965 sentences for development, and 2052 sentences for test. Because we have limited computational power to run the experiments, in order to test and compare different hyper-parameter settings and languages, we have to shrink our datasets. Moreover, the main goal of this thesis is to verify if the BPE embedding can improve the accuracy of POS tagging task based on BI-LSTM-CNNs-CRF architectures. Therefore, when we use the same datasets to test the different neural network architectures, the comparisons are statistically reliable as long as the datasets are sufficiently large.
5 Conclusion and Further Work

In this thesis, we jointly use word-level, subword-level, and character-level representations to perform English, Spanish, and German POS tagging tasks based on the fundamental neural network architecture: BI-LSTM-CNNs-CRF. The main contribution includes (1) proposing an end-to-end system using CNNs to leverage the character- and subword- level knowledges collaborating with word-level features, (2) finding out the pattern of optimal combination of window size and BPE merge operations which achieves improvement over other baseline systems, (3) improving the accuracy of POS tagging on more morphological languages such as Spanish and German than English, (4) through testing the out-of-vocabulary subsets, we found that our model obtains statistical significance improvement on more morphological languages which is not observed on English. (5) based on the error analysis, we found that there are different mislabelling patterns for different languages. There are two interesting part can be further developed. First of all, tuning the best combination of window size and BPE merge operation will be crucial for different morphological languages because these two parameters contribute a significant effect to the results. Secondly, it would be interesting to analyse the errors with detailed examples for German and Spanish.
Figure 5: Confusion Matrix of OOTV for the English testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
Figure 6: Confusion Matrix of OOBV for English testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
Figure 7: Confusion Matrix of OOTV for Spanish testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
Figure 8: Confusion Matrix of OOBV for Spanish testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
Figure 9: Confusion Matrix of OOTV for German testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
Figure 10: Confusion Matrix of OOBV for German testing dataset. The upper table is the confusion matrix of BI-LSTM-CNNs-CRF-BPE, and the lower table is the confusion matrix of BI-LSTM-CNNs-CRF. The column names represent the real tags and the rows represent predicted ones.
References


