Cross-lingual Transfer Learning for Low-Resource Natural Language Processing Tasks

Institute for Anthropomatics and Robotics

Karlsruhe Institute of Technology

Master Thesis

First Supervisor: Prof. Dr. Alexander Waibel

Second Supervisor: Prof. Dr. Tamim Asfour

Advisor: Dr. Thanh-Le Ha

Jinshi Wang

Feb 2021, Karlsruhe

Abstract

As artificial intelligence has gradually become a hot research field in recent years, natural language processing has subsequently become an important topic of discussion. At the same time, neural networks also shine in this field. With the help of various new structures and technologies, the speed and accuracy of neural network dealing with different natural language processing tasks have been greatly improved. But for some languages, when the corpus resources that can be used for training are very scarce, the traditional training methods appear to be stretched. Among the various newly proposed models and methods, the Bidirectional Encoder Representations from Transformers (BERT) model proposed by Google in 2018 has greatly refreshed the accuracy in the NLP field. It can be said that it is the most breakthrough technology from the residual network in recent years. A major feature of BERT is the use of Transformer as the main framework of the algorithm. Transformer can more thoroughly capture the two-way relationship in the sentence, which is very important for our goal. By treating the multilingual BERT as a pre-trained model, for low-resource language to adapt cross-lingual transfer learning. We will use English, German, Chinese and Korean as example. For Natural Language Processing (NLP), Named Entity Recognition (NER) and Sentiment Analysis (SA) are two important aspects of the practical application. In this article, we will discuss the feasibility of different methods based on experimental data, and prove that these methods really help to improve the performance of NER and SA.

Acknowledgement

Many people have provided important help in my writing process, including the professor, my tutor, my friends and my parents.

First, I want to extend my heartfelt gratitude to Professor Waibel and Professor Asfour for leading the extraordinary research team at KIT. It is my honor to have worked with these excellent people in this laboratory.

Then, I would like to give my sincere gratitude to Dr.Thanh-Le Ha, my tutor who gave me a lot of patience and encouragement, provided me with valuable suggestions and ideas, including necessary references and materials, and provided me with great help.

Special thanks to my friends and Jinshu Jiao, who gave me help and encouragement during the difficult time.

Finally, thanks to my parents for their support to my studies over the years and the teaching in life.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Signed : ____

*Date :*____

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years, as artificial intelligence has become more and more popular in academic and industrial, its sub-fields have also achieved considerable development. Natural language processing and artificial neural networks are two very important research areas.

Artificial neural networks, usually simply called neural networks(NN). Neural network is a kind of calculation model, because it imitates the processing mode of human brain neurons, it can easily handle many nonlinear problems that traditional computers are difficult to solve.

Natural language processing, short as NLP, is a bridge between machine language and human language to realize the purpose of human-computer communication. There are many different tasks in NLP, among which Named entity recognition and Sentiment analysis are two very basic tools for many NLP tasks such as information extraction, question answering systems, syntax analysis, and machine translation.

After a long time of various research efforts, we found that using neural networks can achieve very good results when dealing tasks such as named entity recognition(NER) and sentiment analysis. But for both tasks, the existing neural network rely on large annotated data, and performs bad when the language is low-resource or even no data at all, which is difficult to achieve in practice for many languages. In order to solve the above problems, in this work, we propose a cross-lingual model which can transfer the NER and sentiment analysis knowledge from pre-trained multilingual models. Our model will test two NLP tasks in 4 languages to verify the feasibility of this idea.

1.2 Basic Idea

In this work, we will explore the model in two natural language processing tasks mentioned earlier: Named entity recognition and Sentiment Analysis, in four languages:

- English
- German
- Chinese
- Korean

Because of the multilingual BERT is a pre-trained model which trained on a monolingual corpus with 104 languages, so we think we can use the fine-tuned model on one language to evaluate another language, which can be seen as the low-resource language.

In order to ensure if our new model has improvement, we will first set a baseline model only with Bi-LSTM. Then we will fine-tune the pre-trained multilingual BERT model and evaluating on other languages and test the cross-lingual performance of the model. Which can be seen as a cross-lingual transfer learning for low-resource NLP task. The experiments will be done in four languages and two tasks.

In the following chapters, we will first introduce some basic background knowledge. Including the history and development of artificial neural network, the introduction of the basic idea of named entity recognition and sentiment analysis. And the concrete analysis of the newly introduced attention mechanism and BERT model. The idea of cross-lingual representation and transfer learning will also be mentioned in the background. In the next chapter, we will show the concrete model and approach and analyse the advantages. Then the details of the experiments will be listed, including experiments settings and the model structure, also the analysis for each results. The unfinished experiments will also be mentioned here, and future experimental directions will be proposed. In the last chapter, I will show my summary of the entire research.

Chapter 2

Background

2.1 Artificial Neural Networks

2.1.1 Perceptron

Artificial Neural Network(ANN), abbreviated as Neural Network(NN), is a mathematical model or calculation model that imitates the structure and function of biological neural network. The neural network is calculated by connecting a large number of artificial neurons, which we called perceptron. The perceptron consists of 4 parts, input value; weight and bias; net input function; activation function.

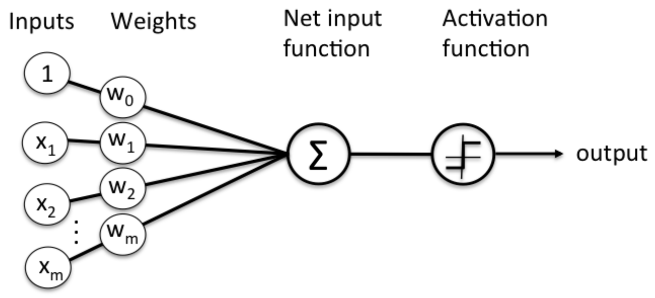


Figure 2.1: The structure of a Perceptron

It has several input terminals that represent the signals received from the input or other perceptrons, and then sum these signals to output or transmit to other perceptrons through the excitation function. This is equivalent to the dendrites and axons of nerve cells. Some neurons are tightly connected, and some are loose. The performance on the perceptron is the weight of each connection edge. These weight information is stored and learned by the neural network. ”Knowledge” place. Here in the showed figure, all the input x_m are multiplied with their weights w_m, and we define weighted sum here which means the sum of multiplied values here, at last apply that weighted sum to the correct activation function. The calculation process can be described as:

$$Output = f\left(\sum_{i=1}^{m} x_i w_i + w_0 x_0\right) \tag{2.1}$$

The perceptron is composed of two layers of neurons. The input layer receives external input signals and transmits them to the output layer. The output layer is M-P neurons (McCulloch-Pitts Neuron). So it is obvious that the perceptron is more suitable for dealing with linear problems. In 1985, Hinton et al.[2] used multiple hidden layers to replace the original single feature layer in the perceptron to solve the non-linear separability problem, and use Back-propagation algorithm(BP) to calculate network parameters. The neurons in each layer are fully interconnected with the neurons in the same layer, and there is no cross-layer connection. Such a neural network structure is usually called a ”Multi-layer Feed-forward Neural Network”.

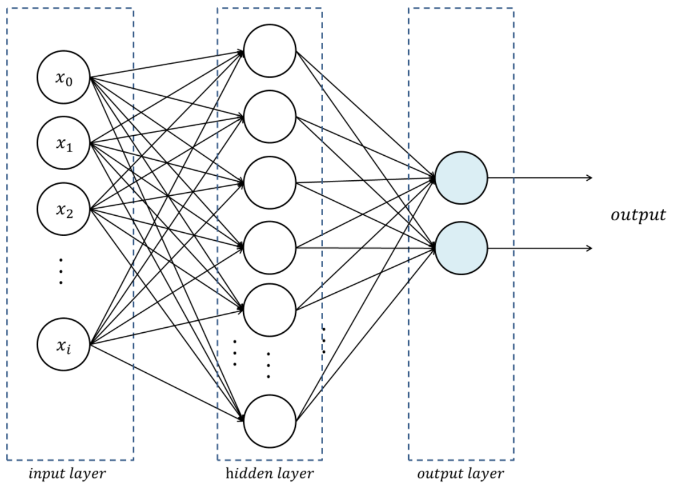


Figure 2.2: The structure of a multi-layer Perceptron

2.1.2 Convolutional Neural Network

The first convolutional neural network was the Time Delay Neural Network (TDNN) proposed by Alexander Waibel et al. in 1987 [3]. TDNN is a convolutional neural network applied to speech recognition problems. It uses FFT preprocessed speech signals as input. Its hidden layer is composed of two one-dimensional convolution kernels to extract translation invariant features in the frequency domain. Because before the emergence of TDNN, the field of artificial intelligence has made breakthroughs in the research of Back-Propagation, so TDNN can use the BP framework for learning. In the original author's comparative experiment, the performance of TDNN surpassed the Hidden Markov Model (HMM) under the same conditions, which was the mainstream speech recognition algorithm in the 1980s.

CNNs are influenced by the time-delay neural network(TDNN). Delayed neural networks reduce learning complexity by sharing weights in the time dimension, and are suitable for processing speech and time series signals. As the first learning algorithm to truly successfully train a multi-layer network structure. CNN uses spatial relationships to reduce the number of parameters that need to be learned to improve the training performance of general forward BP algorithms. In CNN, a small part of the image (local receptive area) is used as the input of the lowest layer of the hierarchical structure, and the information is transmitted to different layers in turn, and each layer passes a digital filter to obtain the most significant features of the observed data. This method can obtain the salient features of observation data that are invariant to translation, scaling, and rotation, because the local receptive area of the image allows neurons or processing units to access the most basic features, such as directional edges or corners. Like showed in Figure 2.3, LeNet-5 proposed by Lecun et al[4] is a typical CNN structure.Until now, CNN is also one of the best methods for feature extraction in the image domain. Many people have also used CNN in natural language processing and have achieved enlightening results.

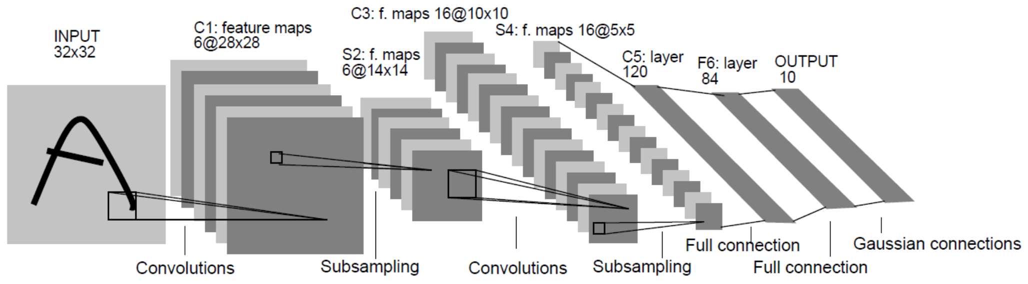


Figure 2.3: Architecture of LeNet-5, a Convolutional neural network

2.1.3 Recurrent Neural Networks

As introduced in the previous chapter, the Recurrent Neural Network(RNN) are designed to work with sequence prediction problems. We can divide sentence prediction problems into the following types: [5]:

- One-to-Many: Single input corresponds to a sequence with multiple outputs.
- Many-to-One: Map one sequence with many steps into a single output like class or prediction.
- Many-to-Many: Which we usually called sequence-to-sequence problem. A synthesis of the previous two questions. One sequence with many steps into a sequence with many steps.

The state of RNN is not only dependent on the input, but also related to the state of the network at a moment. It can be seen that it is different from the feed-forward neural network structure of CNN in that: the structure of feed-forward neural network is directed acyclic graph(DAG), and the structure of recurrent neural network has at least one ring. We assume that the state transition of h occurs in the time dimension, the timeline can be expanded into the following form:

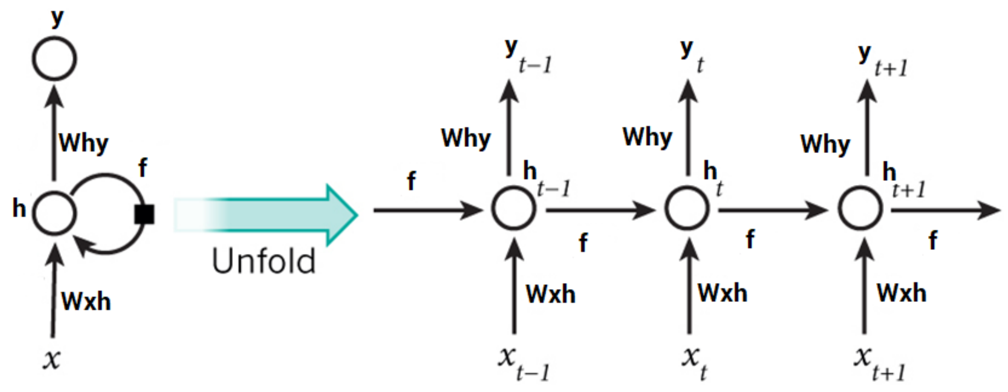


Figure 2.4: RNN timeline expansion diagram

The meaning of the mark in the figure is:

- Circles or squares represent vectors.
- An arrow represents a transformation of the vector. As shown in the figure above, x_{t-1} and f are connected by an arrow, which means x_{t-1} and f have been transformed once.

So we can write its specific expression:

$$h_t = W_{xh}x^t + fh_{t-1} + b_h \tag{2.2}$$

$$o_t = W_{hy}h_t + b_o \tag{2.3}$$

whichx_t means the input at time t,o_t means the output at time t, and$b$ are the bias.

2.1.4 Long Short-Term Memory

But basic RNN also has some obvious disadvantages. Because ordinary RNN is difficult to capture the long-term dependence in the sentence, once the long-term dependence is required, the gradient will disappear. Therefore, Long short-term memory(LSTM) was proposed to solve this problem[6]. One thing to note is that LSTM can only greatly alleviate the disappearance of the gradient of RNN, but it cannot be solved fundamentally. Fortunately, experiments in most task scenarios show that LSTM can achieve good results. The figure above is the internal structure

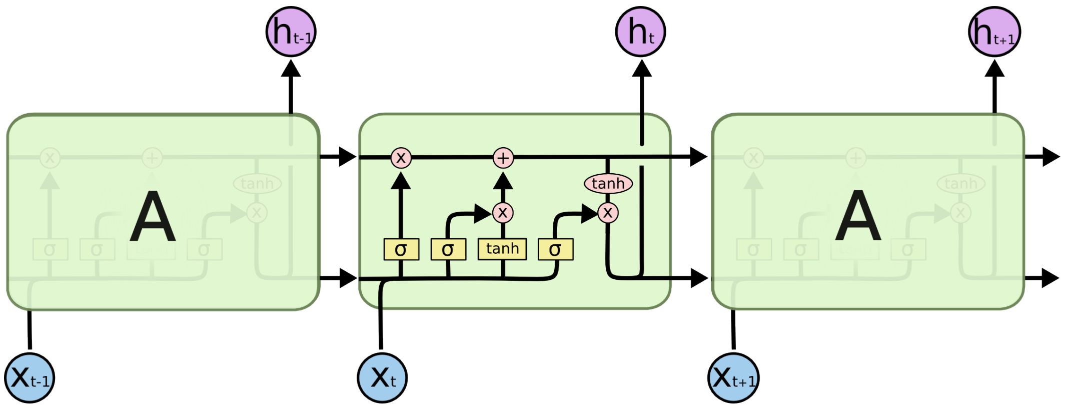


Figure 2.5: The detailed structure of a LSTM cell

of an LSTM cell. There are three main steps inside the LSTM:

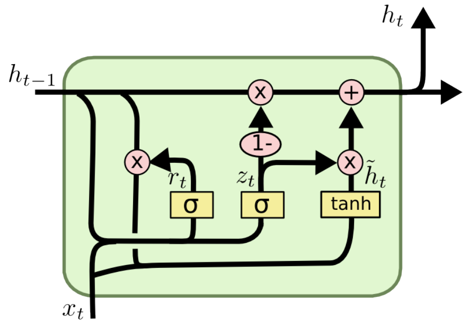
- forget gate: This gate is mainly to selectively forget the input from the previous node. Specifically, the calculated f_t is used as the forget gate to control the c_{t-1} of the previous state which needs to be left and which needs to be forgotten.
- input gate: Some of this is also called update gate. The function of the input gate is to determine which values need to be updated when the cell state is updated.

- output gate: Decide which values in the cell state should be output (that is, the value of the next cell state).

This structure has also achieved good results in many other fields. However, it has a problem in that: the input sequence will be encoded into a fixed-length vector representation regardless of its length, and decoding is limited by the fixed-length vector representation. This problem limits the performance of the model, especially when the input sequence is relatively long, the performance of the model becomes very poor(in the ext translation task, the translation quality is poor when the length of the original text to be translated is too long).

Gated Recurrent Unit

Gated Recurrent Unit(GRU) is a very effective variant of the LSTM network, proposed by Cho,et al[7]. It has a simpler structure than the LSTM network, and the effect is also very good, so it is also a very manifold network at present. Since GRU is a variant of LSTM, it can also solve the long dependency problem in RNN networks. GRU combines the forget gate and input gate into a single "update gate". It also merged cell state and hidden state, and made some other changes.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.6: The detailed structure of a GRU cell

2.1.5 Transformer

Since the Attention mechanism was proposed, the sequence-to-sequence model with attention has been improved in various tasks, so the current sequence-to-sequence model refers to a model that combines RNN and attention. Later, Google proposed a Transformer model to solve the sequence-to-sequence problem, replacing LSTM with a full attention structure, and achieved better results in translation tasks.

The Attention Mechanism

The attention mechanism was first applied to the image field, the idea that was put forward in the 1990s. With the development of neural networks, the paper by google mind team [8] made it popular. They used the attention mechanism on the RNN model for image classification, and then achieved good performance. Subsequently, Bahdanau et, al. used a mechanism similar to attention in the paper[1] to perform translation alignment on machine translation tasks at the same time. Their work is regarded as the first to apply the attention mechanism to NLP field. Then the attention mechanism is widely used in various NLP tasks based on neural network models such as RNN/CNN, and the effect seems to be really good. In 2017,Vaswani eh,al. by google machine translation team published[9], used a lot of self-attention mechanisms to learn text representations, which also achieved good performance.

Mechanism of Attention

As mentioned above, we know that in the sequence-to-sequence model, the encode process of the original code model generates an intermediate vector, which is used to store the semantic information of the original sequence. However, the length of this vector is fixed. When the length of the original input sequence is relatively long, the vector cannot store all the semantic information, and the contextual semantic information is restricted, which also limits the understanding of the model. Therefore, the attention mechanism is used to break the limitation of the original encode and model on the fixed vector.

The essence of attention mechanism is to get inspiration from human visual attention mechanism. Roughly when we perceive things visually, we generally don't look at a scene from the beginning to the end every time, but often observe and pay attention to a specific part according to needs. And when we find that something we want to observe often appears in a certain part of a scene, we will learn to focus on that part when similar scenes appear in the future. The principle of attention mechanism is to calculate the matching degree between the current input sequence and the output vector. The higher the matching degree, the higher the relative score of the attention point. Among them, the matching degree weight calculated by attention is limited to the current sequence pair, not the overall weight like the network model weight.

As shown below [1], the encoder is a bidirectional RNN with a forward hidden state $\overrightarrow{h_i}$ and a backward one $\overleftarrow{h_i}$. Here the encoder generates h_1,h_2,...,h_T from the input x_1,x_2,...,x_T. All the vectors h_1,h_2,...,h_T what is used in the work is basically the combination of the forward and backward hidden states.

$$h_j = \left[\overrightarrow{h}_j^T; \overleftarrow{h}_j^T\right] \tag{2.4}$$

The context vector c_i for the output word y_i is like follow:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.5}$$

The weights α_{ij} are computed as follow:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.6}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.7}$$

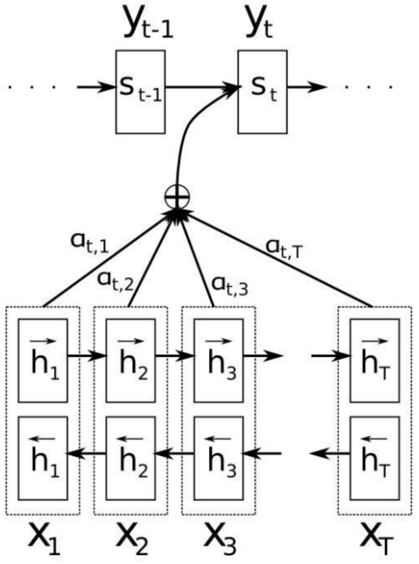


Figure 2.7: Given a source sentence, how to generate the t-th target word[1]

The key operation here is to calculate the weight of the correlation between the encoder and the decoder state to obtain the attention distribution, so as to obtain the weight of the more important input position for the current output position, which will account for a larger proportion when predicting the output. Through the introduction of the attention mechanism, we can broken the limitation of only using the final single vector result of the encoder, so that the model can focus on all the input information that is important for the next target word, and the model effect is greatly improved. Another advantage is that by observing the changes in the attention weight matrix, we can better know which part of the translation corresponds to which part of the source text, which helps to better understand the working mechanism of the model.

Attention Mechanism Classification

In 2015, Luong et,al. published a very representative paper[10]. This study told everyone how attention can be extended in RNN, which is useful for subsequent applications of various attention-based models in NLP. In the paper, they proposed two attention mechanisms:

- Global Attention: The same as the traditional attention model. All hidden states are used to calculate the weight of the context vector, that is, the variable-length alignment vector at whose length is equal to the length of the input sentence on the encoder. The structure is shown in the figure below.
- Local Attention: One obvious shortcoming of global attention is that every time, all hidden states on the encoder must participate in the calculation. This calculation overhead will be relatively large, especially when the encoder sentence is too long.

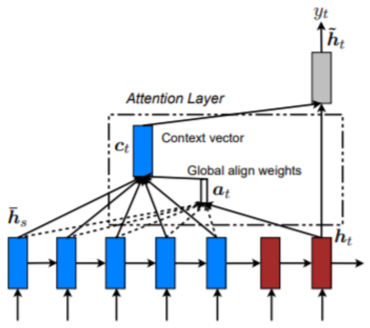


Figure 2.8: Global attention structure

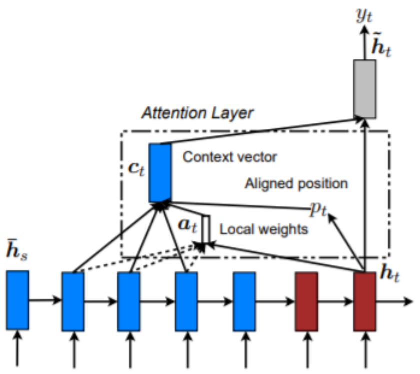


Figure 2.9: Local attention structure

Transformer model

In 2017, vaswani et al. of Google's machine translation team published the paper[9] and proposed the transformer model. Transformer's performance on machine translation tasks surpasses RNN and CNN. It can achieve very good results with only encoder-decoder and attention mechanisms. The biggest advantage is that it can be efficiently parallelized. The model has two main highlights:

- Different from the previous mainstream machine translation using the RNN-based sequence-to-sequence model framework, this paper uses the attention mechanism instead of RNN to build the entire model framework.
- A multi-head attention mechanism method is proposed, which uses a large number of multi-head self-attention mechanisms in encoders and decoders.

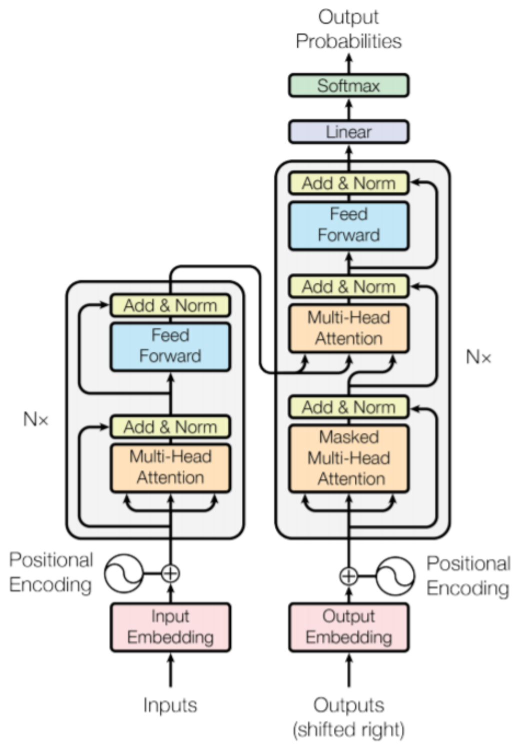


Figure 2.10: The Transformer-model architecture

Self-attention

The idea of self-attention is similar to that of attention mechanism, in base-Attention, it is mainly to find the attention weight of the words in the source sentence and the words in the target sentence. And Self-Attention is to find an attention value between the word in this sentence and all other words. As shown below, three vectors are created for each input word of the encoder, namely Query vector, Key vector, Value vector, these vectors are obtained by multiplying embedding and three matrices. Take a look at the calculation process of this Self-Attention. As shown below, first we need to get the dot product of Q and K, and then in order to prevent the result from being too large, it will be divided by a scale $\sqrt{d_k}$, where d_k is the dimension of a query and key vector. Then use the softmax function to normalize the result to a probability distribution, and then multiply it by the matrix V to get the weighted summation.

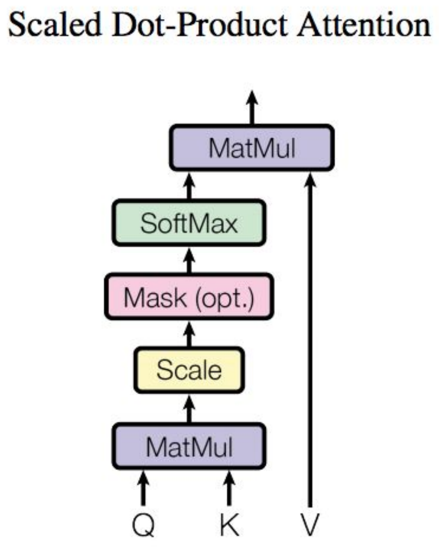


Figure 2.11: Self-attention structure

Multi-Headed Attention

In order to improve the performance of the attention layer, the transformer also proposes a multi-head attention mechanism, which is to initialize not only a set of Q, K, and V matrices, but also multiple sets. In[9] the author used 8 matrices, which means each encoder/decoder will get 8 sets. Specifically, it is to define 8 sets of weight matrices, each word will do the self-attention calculation 8 times. But only one matrix can be received at feed-forward, so through different linear transformations, Q, K, and V are projected, and finally the unreasonable attention results are spliced together. It mainly improve two parts:

- It expands the model's ability to focus on different locations.
- It gives multiple "representation subspaces" of the attention layer.

Positional Encoding

In addition to word embedding, Transformer also needs to use position embedding to indicate the position of the word in the sentence. Because transformer does not use the structure of RNN, but uses global information, it cannot use the order information of words, and this part of information is very important of NLP. Therefore, the position embedding is used in transformer to save the relative or absolute position of the word in the sequence.

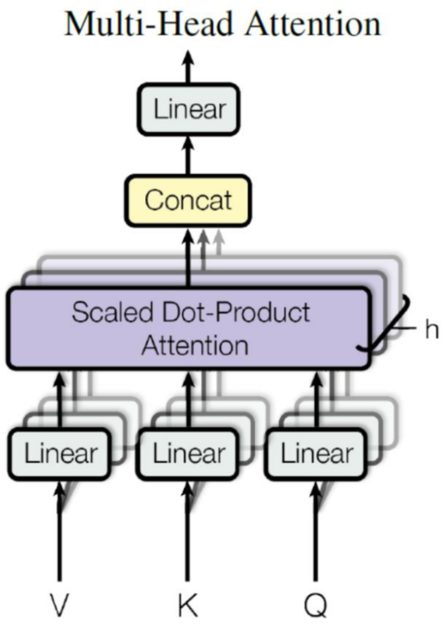


Figure 2.12: Multi-head attention structure

The position embedding is represented by PE, and the dimension of PE is the same as the word embedding. PE can be obtained through training or calculated using a certain formula as follow:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{2.8}$$

$$PE_{(pos,2i)} = cos(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{2.9}$$

2.2 Natural Language Processing

2.2.1 Named Entity Recognition

Named Entity Recognition (NER) is an important basic tool in application fields such as information extraction, question answering systems, syntax analysis, and machine translation, and it occupies an important position in the process of natural language processing technology becoming practical. Named entities generally refer to entities with specific meaning or strong referentiality in the text, usually including names of persons, places, names of organizations, dates and times, proper nouns, etc. The NER system extracts the above entities from unstructured input text, and can identify more types of entities, such as product names, models, prices, etc., according to business needs. Therefore, the concept of entity can be very broad, as long as it is a special text fragment required by the business, it can be called an entity. NER is a fundamental and key task in NLP. From the perspective of natural language processing, NER can be regarded as a type of unregistered word recognition in lexical analysis. It has the largest number of unregistered words, and the greatest impact on word segmentation. At the same time, NER is also the basic for many NLP tasks such as relationship extraction, event extraction, knowledge graphs, machine translation, and question answering systems. In the early days, we always using dictionary and rule-based approach to deal with NER tasks, which rely on a large number of manual annotations and corpora. After that, many methods based on traditional machine learning were proposed. In recent years, with the development of artificial intelligence, a large number of methods based on neural networks have also been proposed.

Traditional machine learning methods: HMM and CRF

In machine learning-based methods, NER is treated as a sequence labeling problem. Use large-scale corpus to learn a labeling model to label each position of the sentence. Commonly used models in NER tasks include generative model HMM, discriminative model CRF, etc.

Hidden Markov model(HMM), is a relatively classic machine learning model. It is widely used in language recognition, natural language processing, pattern recognition and other fields. In HMM, there are 5 basic elements: {N,M,A,B,Π}. In NER task, we will introduce these 5 basic elements in combination with the sequence marking task:

- N: A limited set of states. Here, it refers to the label behind each word.
- M: A finite set of observations. Here, it refers to each word itself.
- A: State transition probability matrix. Here, it refers to the probability of transferring from one label to the next one.
- B: Observation probability matrix, that is, emission probability matrix. Here, it refers to the probability of generating a certain word under a certain label.
- Π : Initial probability matrix. Here, it refers to the initialization probability of each label.

The above elements can all be counted from the training corpus. Finally, based on these statistical values, we apply the some algorithms to calculate the label sequence behind the word sequence, for example the Viterbi algorithms.

Conditional Random Field(CRF), is one of the mainstream model of NER. Its objective function not only considers the input state characteristic function, but also includes the label transition characteristic function. When the model is known, finding the predicted output sequence for the input sequence is the optimal sequence that maximizes the objective function, which is a dynamic programming problem, which can be decoded by the Viterbi algorithm to obtain the optimal label sequence. The advantage of CRF is that it can utilize rich internal and contextual feature information in the process of marking a location.[11]

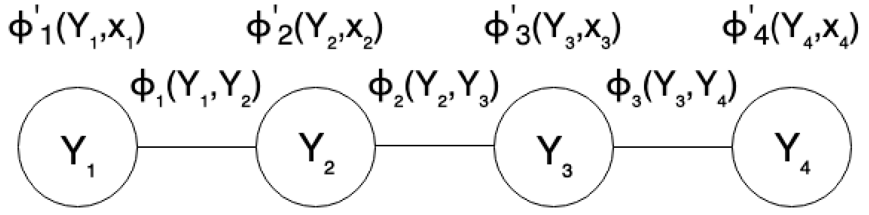


Figure 2.13: CRF model, conditioned on X

LSTM+CRF: Bi-LSTM+CRF

In recent years, with the development of hardware computing capabilities and the introduction of word embedding, neural networks can effectively handle many NLP tasks. This type of method handles sequence labeling tasks (such as CWS, POS, NER) in a similar way: the token is mapped from the discrete one-hot representation to the low-dimensional space to become dense embedding, and then the embedding sequence of the sentence is input to the RNN In, the neural network is used to automatically extract features, and Softmax is used to predict the label of each token. The main components of the Bi-LSTM+CRF model applied to NER:

- Embedding layer: There are mainly word vectors, word vectors and some additional features
- Bidirectional LSTM layer: feature extractor
- The final CRF layer: do sentence-level tag prediction.

Experimental results show that Bi-LSTM+CRF has reached or surpassed the CRF model based on rich features, and has become the most mainstream model of NER based on deep learning. In terms of features, the model inherits the advantages of deep learning methods. It does not require feature engineering. It can achieve good results by using word vectors and character vectors. If there are high-quality dictionary features, it can be further improved.[12]

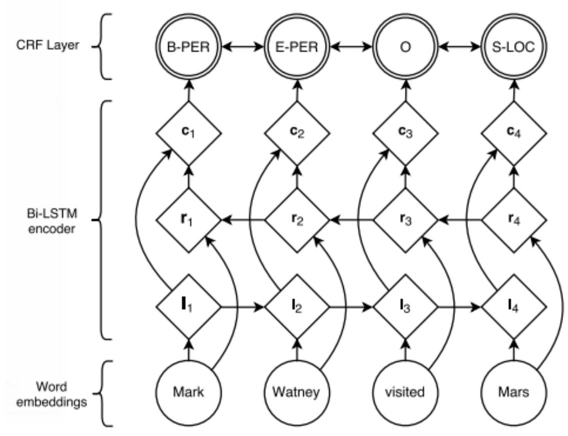


Figure 2.14: Main architecture of the Bi-LSTM+crf network

CNN+CRF: IDCNN-CRF

Although CNN has weaknesses in feature extraction of long sequences, the CNN model has parallel capabilities and has the advantage of fast calculation speed. The introduction of dilated convolution enables CNN to take into account the calculation speed and feature extraction of long sequences in NER tasks.

For sequence labeling, ordinary CNN has a shortcoming, that is, after convolution, the last layer of neurons may only get a small piece of information in the original input data. For NER, each word in the entire input sentence may have an impact on the current position labeling, which is the so-called long-distance dependency problem. In order to cover all the input information, more convolutional layers need to be added, resulting in deeper and deeper layers and more and more parameters. In order to prevent over-fitting, more regularization such as Dropout must be added, which brings more hyper-parameters, and the entire model becomes large and difficult to train. Because of the disadvantages of CNN, people still choose a network structure such as Bi-LSTM for most sequence labeling problems, and use the memory of the network to remember the information of the whole sentence as much as possible to label the current word. But this brings another problem. Bi-LSTM is essentially a sequence model, which is not as powerful as CNN in the use of GPU parallel computing.

In 2015 Fisher Et al.proposed the dilated CNN model[13], which means ”expanded” CNN. The idea is not complicated: the filters of normal CNN all act on a continuous area of the input matrix, continuously sliding to do convolution. The dilated CNN adds a dilation width to the filter. When it is applied to the input matrix, it will skip all the input data in the middle of the dilation width; and the size of the filter itself remains the same, so that the filter obtains the data on the wider input matrix. It looked like it was ”expanded”. Connecting the CRF layer to the end of network models such as Bi-LSTM or IDCNN is a very common method of sequence labeling. The Bi-LSTM or IDCNN calculates the tag probability of each word, and the CRF layer introduces the transition probability of the sequence, and finally calculates the loss and feeds it back to the network.

BERT+(LSTM)+CRF: BERT implementation

BERT contains a lot of general knowledge. Using a pre-trained BERT model and using a small amount of labeled data for Fine-tuning is a fast way to obtain a good NER. First obtain the BERT pre-training model and modify the data pre-processing code, build the model and then add Bi-LSTM and CRF layers for decoding. The concrete implementation will be introduced in the later chapter.

2.2.2 Sentiment Analysis

Sentiment analysis is a common application of natural language processing (NLP) methods. It is a method of analyzing, processing, summarizing and inferring sentimental subjective text, using some sentiment score indicators to quantify qualitative data. In natural language processing, sentiment analysis is a typical text classification problem, that is, the text that needs sentiment analysis is divided into its categories. Now there are two mainstream sentiment analysis methods: one based on dictionary and the other based on Methods of machine learning algorithms. The dictionary-based method mainly uses a series of emotional dictionaries and rules to disassemble the text, extract keywords, calculate the emotional value, and finally use the emotional value as the basis for the emotional tendency of the text. This method often ignores the word order, grammar and syntax, and treats this text as just a word set, so it cannot fully express the semantic information of the text.

Method based on Sentiment Dictionary

The dictionary-based method mainly uses a series of emotional dictionaries and rules to disassemble, analyze and match the text to the dictionary (generally with part of speech analysis and syntactic dependency analysis), calculate the emotional value, and finally use the emotional value as the emotional tendency of the text The basis for judgment. The general steps of dictionary-based sentiment analysis are as follows:

- Disassemble sentence operations on text larger than sentence granularity, with sentence as the smallest unit of analysis
- Analyze the words that appear in the sentence and match them according to the emotional dictionary
- Dealing with negation logic and transition logic
- Calculate the emotional word score of the whole sentence (weighted summation based on different words, different polarities, and different degrees)
- The sentiment orientation of the sentence is output according to the sentiment score.
- If it is a sentiment analysis task at the text or paragraph level, according to the specific situation, it can be carried out in the form of a single sentiment analysis and fusion of each sentence, or it can first extract the sentiment topic

sentence and then perform the sentiment analysis of the sentence to obtain the final sentiment analysis result.

Method based on Machine Learning

The dictionary-based method is convenient and efficient to implement. But the main disadvantage is that reliable dictionaries are hard to find, and the information found is basically a specific field. In addition, due to the ever-changing nature of natural language, the rules are difficult to satisfy most scenarios. There is a certain gap between the effect and the effect of learning-based methods.

With the development of machine learning, the current machine learning algorithms for sentiment analysis are relatively mature. Deep learning is suitable for text data processing and semantic understanding. Deep learning has a flexible structure, and its bottom layer uses word embedding technology to avoid processing difficulties caused by uneven text length. Using deep learning abstract features can avoid a lot of manual feature extraction work. Deep learning can simulate the connection between words, and has the function of abstraction of local features and memory. It is these advantages that make deep learning play a pivotal role in sentiment analysis and even text analysis and understanding.

Currently, deep learning neural networks used in sentiment analysis include multi-layer neural networks (MLP), convolutional neural networks (CNN), and long and short memory models (LSTM). These neural networks have many parameters in common, and they need to be updated through backward propagation. CNN and LSTM, as different types of neural network models, require relatively few parameters, which also reflects their commonality: parameter sharing. This is similar to the traditional machine learning principle: the more restrictions on the parameters or the model, the less the degree of freedom of the model, and the less likely it is to overfit. Conversely, the more model parameters, the more flexible the model and the easier it is to fit noise, which will have a negative impact on prediction. The optimal parameters (for example, several layers of models, the number of nodes in each layer, dropout probability, etc.) are usually selected through cross-validation techniques. Sentiment analysis is essentially a classification problem, a type of supervised learning. Other classic machine learning models, such as SVM, random forest, logistic regression, etc., are compared with neural network models.

2.3 Transfer learning

2.3.1 Cross-lingual Representation

After Mikolov et al.[14] formally proposed the vector representation of words (word embedding) in 2013, word embeddings have become a cornerstone of the NLP field. At the same time, sentence embeddings are also proposed, sentence embedding performs well in some NLP tasks, but when a lot of information is integrated into one vector, the information will be lost[15], so it is performed in most tasks not good. However, word embeddings of different languages cannot be directly used in multilingual or cross-lingual NLP applications. Due to the different richness of corpus resources available in different languages, researchers will naturally consider how to transfer the knowledge learned in resource-rich languages to languages with relatively poor resources. In this case, research work on cross-language representation is gradually emerging.

Cross-lingual Word Embedding

Cross-lingual word embedding refers to word embeddings of different languages into the same word vector space, so that words with the same meaning but from different languages have the same vector representation. Through the research of Sebastian et al.[16], we found that the effect of cross-language word embedding mainly depends on the different requirements for the data, not the structure of the model. Therefore, the classification standard is not according to the type of model, but according to the data requirements. Data requirements can be classified in two dimensions:

- The way of alignment: Can be divided into 3 types, word alignment, sentence alignment and article alignment
- The degree of similarity of aligned words and sentences. Can be divided into, parallel data and similar data

Representation from BERT

The BERT can take one or two sentences as input, and take special token [CLS] and [SEP] as the start and end of a sentence.

For tokenization, BERT will have a WordPiece process during data preprocessing. WordPiece is literally understood as breaking word into pieces. One of the main implementations of WordPiece is called Byte-Pair Encoding(BPE) double-byte encoding. For example, the three words ”loved”, ”loving”, and ”loves”. In fact, the semantic meaning of itself is the meaning of ”love”, but if we use words as the unit, then they are not the same words. There are many words with different suffixes in English, which will make the vocabulary become very large. The speed slows down, and the training effect is not very good.

After tokenization, all the tokens will be converted to an ”id” as present in tokenizer vocabulary. And last pad to the same max size.

2.3.2 Transfer learning

The transfer learning is to transfer the trained model to similar tasks to help the training of the new model. An obvious advantage is that the use of transfer learning can reduce the amount of training data of the model and speed up the training speed. Transfer learning is mainly divided into two steps:

- Use some large public data sets to train a ”universal” model. This stage is called the pre-training stage.
- Use these pre-trained public models to fine-tune the model for our task. This stage is also called the fine tuning stage.

Driven by some AI giants, more and more people have invested in this research and have produced a series of research results such as EMLo, UMLFIT, GPT, BERT, etc., and achieved certain results. In our work, we will focus on the BERT which achieve best results among them.

BERT

Bidirectional Encoder Representations from Transformers(BERT) was proposed by google in 2018[17], as a replacement for word-to-vector, it has greatly refreshed its accuracy in 11 directions in the NLP field. It can be said that it is the best break-through technology of self-residual network in recent years. The main features of the paper are as follows:

- Using transformer as the main framework of the algorithm, transformer can more thoroughly capture the two-way relationship in the sentence;
- Using the multi-task training goals using Mask Language Model(MLM) and Next Sentence Prediction(NSP);
- The use of more powerful machines to train larger-scale data makes the results of BERT reach a whole new level.
- Another obvious structural difference of BERT is to change the common one-way network structure to two-way. Google believes that the meaning of a word is not enough to use the above information, but the following is still needed.

Mask Language Model

BERT uses a two-way language model, but if this is two-way, you cannot use the method of predicting the next word, because the model will see the value to be predicted. So BERT used the mask language model (MLM) task for the first time. The core idea of the Masked Language Model(MLM) is taken from a paper published by Wilson Taylor in 1953[18]. The so-called masked language model refers to masking some words from the input expected during training, and then predicting the word through the context. Just as the traditional language model algorithm matches RNN, this property of MLM matches the structure of Transformer very well. There are two types of masks in the Transformer model:

- Padding Mask: Processing non-fixed-length sequences, distinguishing padding and non-padding parts, such as the application of RNN and other models and Attention mechanism.
- Sequence Mask: Prevent label leakage, such as: mask matrix in Transformer decoder, [Mask] bit in BERT, mask matrix in XLNet, etc.

Next Sentence Prediction

Many important downstream tasks, such as intelligent question answering (QA) and natural language inference (NLI), are based on understanding the connection between two sentences, and standard language models cannot directly capture this connection. BERT proposed a binary next sentence prediction task for pre-training, which can be implemented on any monolingual corpus. Specifically, when selecting sentence pairs A and B for each training sample, there is a 50% probability that B is the true last sentence of A, 50% probability that it is a random sentence in the corpus.

Input of BERT

The input part of BERT is a linear sequence, two sentences are separated by a separator, and two identifiers are added at the first and last. There are three embeddings for each word:

- Token Embedding: This is the most important information about the word in the model, usually known as word vector. In BERT, in order to facilitate identification, we set the first position of all embedding to [CLS] for subsequent processing.
- Segment Embedding: Because of the prediction task of the next sentence in BERT, there will be two sentences spliced together, the previous sentence and

the next sentence, the previous sentence has the upper sentence vector, and the next sentence has the next sentence vector. In addition, [SEP] is added to the end of the sentence, and [CLS] is added to the beginning of the two sentences.

- Position Embedding: The word order in NLP is a very important feature, the embedding vector learned. This is different from Transformer, which has a preset value.

Chapter 3

Approaches

Cross-lingual learning[19] has been a very popular research topic in NLP for a long time. But always requires large bilingual dictionaries or parallel data. So in order to make progress in the low-resource NLP task, recent work has begun to focus on cross-lingual transfer learning. Our work will also focus on this goal based on the pre-trained multilingual model. In the chapter, we will first introduce some related previous studies, and the details of our model will be described.

3.1 Related Works

So far, many people in the field cross-lingual learning have done a lot of research. Every year, there are many new ideas for multilingual transfer learning applications, researchers and business fields have made a lot of efforts and practices.

We can get some inspiration from many articles of predecessors. Both in cross-lingual evaluation and low-resource transfer learning. Despite many positive results, various experiments have also shown many limitations of the language model.

- Early in 2016 Ha et al.[20] proposed an Universal Encoder and Decoder machine translation framework which showed that integrated monolingual corpus or other low-resource parallel corpus into a shared representation vector can significantly improve the performance of the model.
- Later, Hsu et al.[21] test the cross-lingual zero-shot reading comprehension with multi-BERT, which shows using the pre-trained model on zero-shot learning is feasible.
- Pires et al.[22] made a large number of probing experiments, to examine the properties of mBERT on zero-shot language in Named Entity Recognition and

part-of-speech Tagging tasks. But it shows the typological similarity is a very important influencing factor of zero-shot transfer.

- Wang et al.[23] proposed an approach for contextualized cross-lingual embedding learning, which get a strongly convinced result in zero-shot dependency parsing. However, the representation used for the parser was a bilingual projection, and the projection was based on word embeddings trained on parallel data.
- Similarly, Wu et al.[24] tried to look into the cross-lingual potential of mBERT, when training as a zero-shot language transfer model on multi-NLP tasks in 39 languages. And shows the relationship of zero-shot languages in both type-level and token-level.
- XTREME, by Hu et al.[25], is a recently benchmark for multilingual representation evaluation, which in natural language inference and question answering did comparison between XLM-R and BERT.
- Liu et al.[26] compared non-contextualized word embeddings, and shows the the amount of training data is critical for alignment, which is a important information for the mBERT cross-lingual ability.
- Libovickỳ et al.[27] examines on the language-neutrality of mBERT with respect to lexical semantics.

The above studies have done a lot of research on M-BERT in different tasks, and provided me with a lot of help, but there are still some omissions in the exploration of specific NER and SA tasks in specific languages. In addition, there are also some researches on other tasks of M-BERT and input classification. Since our research is only for reference but not inspired, it is not listed here.

3.2 Baseline Model

Since both tasks are classification tasks, we set the basis for classification:

- Named Entity Recognition: for NER task, we use IOB tagging and for the whole experiment using three type: {Person, Location, Organization}, all the tags belongs to the set {O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG}.
- Sentiment Analysis: for SA task, we choose 2 types of emotion, the data were all for comment or twitter, so with the set {Positive, Negative}.

3.2.1 Model Structure

The baseline model of NER task is character-based Bi-LSTM+CRF sequence labeling model. Similar to NER, because text classification only requires tags relative to the entire sentence, so the result is directly expanded and then linearly changed. So we removed the CRF layer of the NER baseline model. The model is similar to the models provided by paper[28] and [29], the structure is showed in Figure 5.1. The sample is showed in Chinese, each Chinese character in the input sentence will be classified into the tag in the classification set.

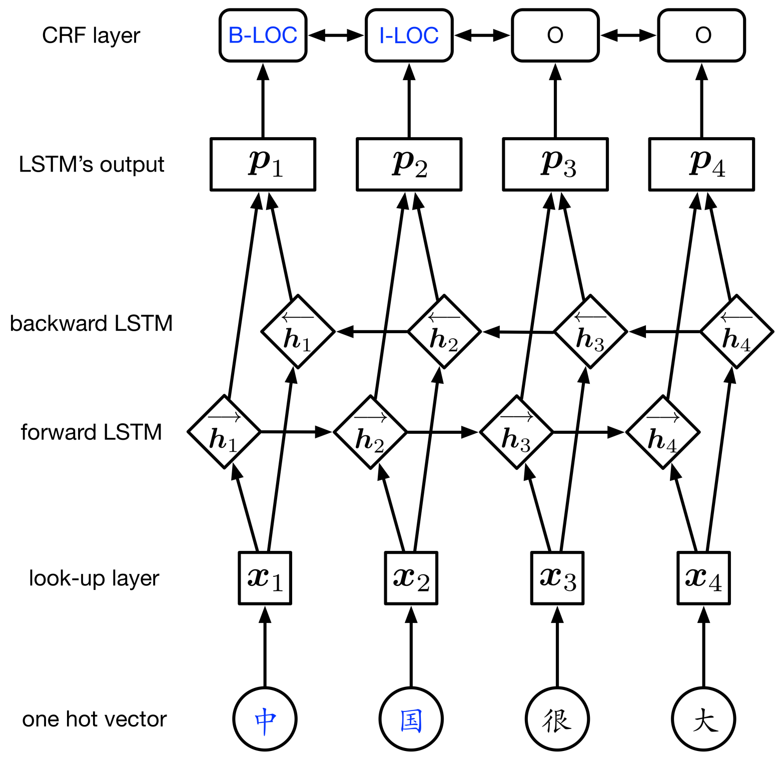


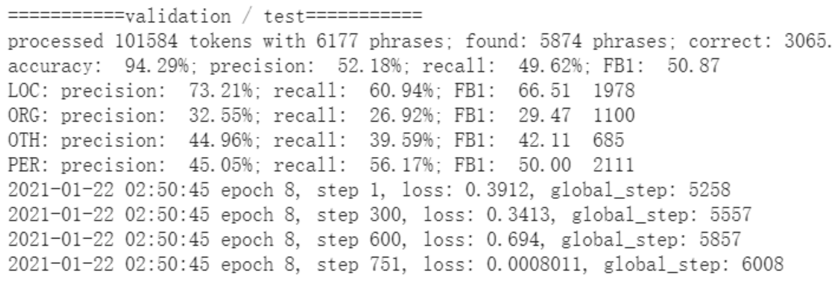
Figure 3.1: Bi-LSTM+CRF model instructure

- The first layer: look-up layer. Here we refer to the research from Lample et al.[29]. Convert each word entered into an embedding matrix through word-embedding. Our approach is to initialize each matrix randomly.

- The second layer: Bi-LSTM layer, can better capture the two-way semantic dependence.
- The third layer: CRF layer. The output of the softmax layer is independent of each other, that is, although BiLSTM has learned the context information, the output has no effect on each other. It just selects a label with the largest probability value for output at each step, and the final label is labeled by each sequence position Splicing, this is only the local optimal solution obtained without considering the overall situation. Therefore, it will lead to irregularities in the obtained label (such as B-person followed by a B-person problem). But after adding CRF, it selects the larger overall probability of the labeling result among all possible labeling results.

3.2.2 Evaluating

For the named entity recognition task, cause we test on the CONLL2003 dataset, so we use the official evaluation script: conlleval.perl. Since all the data in our task is in the form CONLL2003. So the whole NER task we use this tool to get the result. Which the output like this:



```
===========validation / test===========
processed 101584 tokens with 6177 phrases; found: 5874 phrases; correct: 3065.
accuracy:  94.29%; precision:  52.18%; recall:  49.62%; FB1:  50.87
LOC: precision:  73.21%; recall:  60.94%; FB1:  66.51  1978
ORG: precision:  32.55%; recall:  26.92%; FB1:  29.47  1100
OTH: precision:  44.96%; recall:  39.59%; FB1:  42.11  685
PER: precision:  45.05%; recall:  56.17%; FB1:  50.00  2111
2021-01-22 02:50:45 epoch 8,  step 1,  loss: 0.3912,  global_step: 5258
2021-01-22 02:50:45 epoch 8,  step 300,  loss: 0.3413,  global_step: 5557
2021-01-22 02:50:45 epoch 8,  step 600,  loss: 0.694,  global_step: 5857
2021-01-22 02:50:45 epoch 8,  step 751,  loss: 0.0008011,  global_step: 6008
```

Figure 3.2: Evaluating the baseline model using conlleval.perl

Different from NER, in our work sentiment analysis only classify the input into two emotion, positive or negative. So we just get the accuracy of the output.

3.3 Cross-lingual Transfer Learning

As mentioned above, BERT has a very good performance on transfer learning. A year later, Devlin et al.[17] released the multilingual BERT (mBERT), which has the same structure as monolingual BERT, but the data is from Wikipedia in 104 languages. Since mBERT has cross-lingual alignment, which means mBERT outputs word embeddings with the similar meaning in different languages, usually has the close distance in shared vector place. We can fine-tune the pre-trained model in one language, and evaluate on another low-resource language, which treated as the cross-lingual transfer learning.

3.3.1 Transfer learning with BERT

After training the BERT on a large number of orders, it can be applied to various tasks of NLP. For other tasks, we can also make corresponding predictions based on the output information of BERT. The following figure[17] shows the transfer method of BERT in 11 different tasks. We only need to add an output layer on the basis of BERT to complete the fine-tuning of specific tasks, which realizes the transfer learning. In our task, both the NER and Sentiment analysis can be seen as the

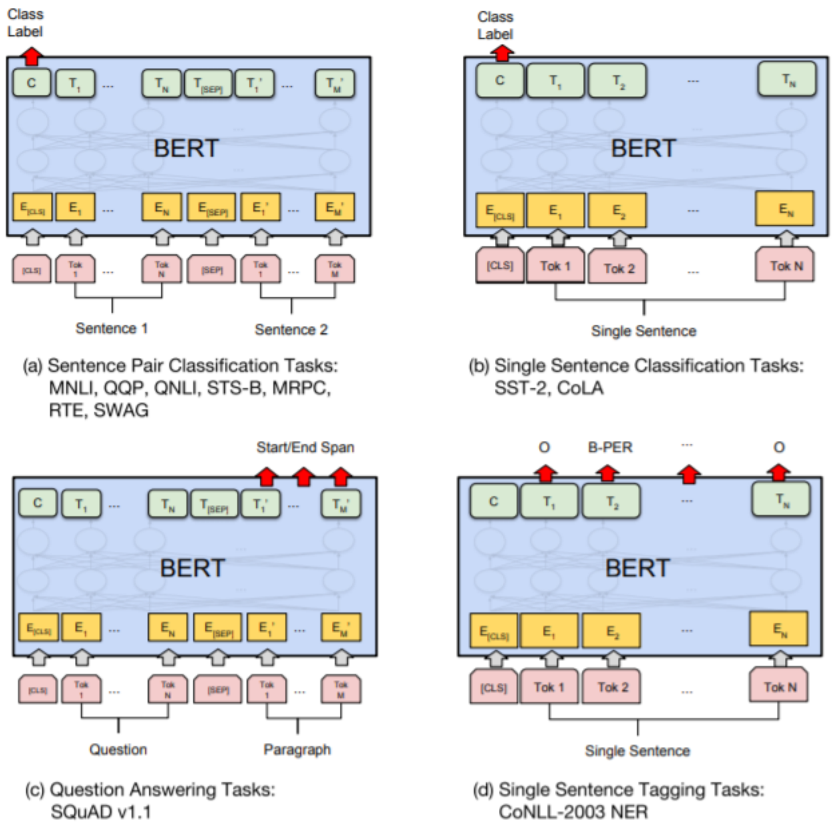

Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 3.3: Illustrations of Fine-tuning BERT on Different Tasks

single sentence tagging task. After the first starting symbol [CLS] passes through the Transformer encoder, the recognition system inputs a text sequence of labeled entity categories for fine-tuning training. When identifying entity categories, each token vector of the sequence send it to the classification layer that predicts the NER label for recognition. There are three types of fine tuning strategies:

- Train the entire model: We can further train on the basis of the pre-trained model and update the parameters and weights of the new model. This will be use when the new data set is relatively large.
- Train some layers and others unchanged: Keeping the bottom layer of the pre-trained model and training the high layer, generally also requires a large amount of new data.
- Maintain the entire pre-trained model: Keep the entire pre-trained model unchanged, and add some neural network layers after its output to adapt to the corresponding tasks, so that only the new network needs to change the weights and parameters.

Since our task does not have a large number of datasets and we want to transfer to a low-resource language task, we use the third fine-tuning strategy.

3.3.2 Multilingual BERT

In order to transfer learning into low-resource tasks, even zero-shot cases. We introduce a new multilingual BERT model. With the same structure like BERT, multilingual BERT uses a 110k shared WordPiece vocabulary, which allows the shared embeddings cross languages like showed in Figure. Therefore, after we fine-tuning our task on a language with large dataset, due to the shared embedding space, we can directly extend the model to a low-resource language. For example, fine-tuning the mBERT on named entity recognition task with English dataset, and then evaluate the Chinese dataset. The whole process can be seen as a cross-lingual transfer learning for low-resource NER task.

3.4 Model structure

The overall structure is as follows:

- Fine tuning the source language task using multilingual BERT.
- Get the output of the hidden state of BERT

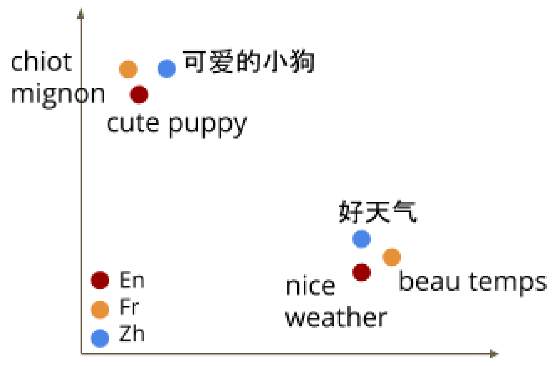


Figure 3.4: Cross-lingual alignment of multilingual BERT

- Send the output into the Bi-LSTM layer for classification task
- Add a CRF layer if dealing the NER and direct get the output if dealing with Sentiment analysis

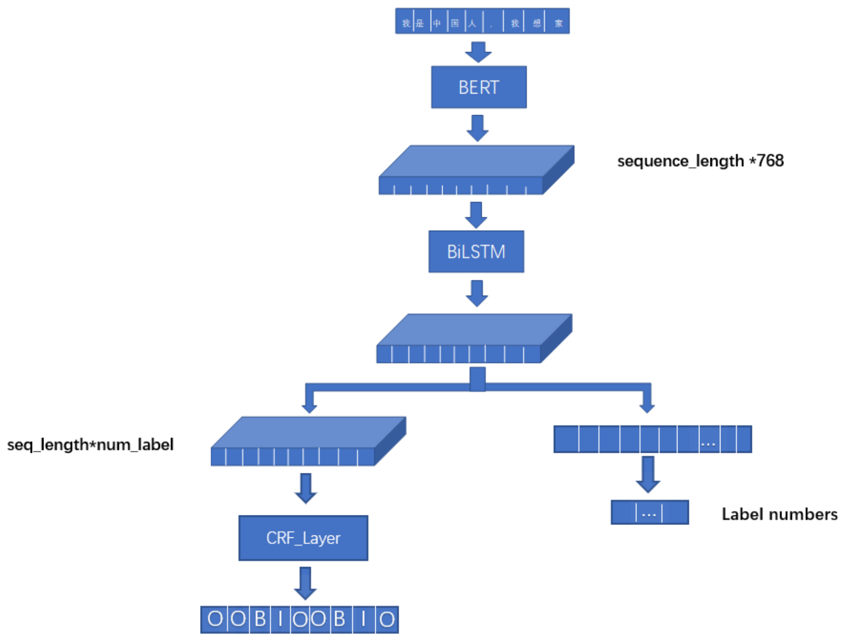


Figure 3.5: The entire forward propagation calculation process with BERT, Bi-LSTM and CRF

3.4.1 BERT model

As mentioned above, because we use the third fine-tuning method. So we just need to adjust the input to adapt the multilingual BERT model. The BERT model has a *processor* class, which we need to make adjustments for each different language task.

3.4.2 Bi-LSTM layer

In tensorflow, there is an advanced package *tf.contrib.rnn.LSTMBlockFusedCell* of the LSTM unit to perform better on CPU, but in this work we use GPU to train, and the space is enough, so we use *tf.contrib.rnn.LSTMBlockCell* and use *tf.concat* to combine forward and backward inference together. The core code part is shown in the figure below.

```
def blstm_layer(self, embedding_chars):

  with tf.variable_scope('rnn_layer'):
    cell_fw, cell_bw = self._bi_dir_rnn()
    if self.num_layers > 1:
        cell_fw = rnn.MultiRNNCell(
             [cell_fw] * self.num_layers,
             state_is_tuple=True)
             cell_bw = rnn.MultiRNNCell(
        [cell_bw] * self.num_layers,
             state_is_tuple=True)
    outputs, _ = tf.nn.bidirectional_dynamic_rnn(
    cell_fw, cell_bw,
    embedding_chars,dtype=tf.float32)
    outputs = tf.concat(outputs, axis=2)
  return outputs
```

There are two things we need to pay attention to here:

- *tf.concat*: When talking about BERT, we mentioned that BERT is a model that can infer the current word from the context in a real sense, but Bi-LSTM cannot. However, the combination of Bi-LSTM is only the result of a simple concat after two one-way inferences. In fact, each inference is still one-way, and the BERT model truly realizes inference through context through the mask method.
- If we use the most basic LSTM unit in Tensorflow, the output does not need to be transposed, and the input shape in the call method of *LSTMBlockFusedCell*

is [time_len, batch_size, input_size], so transpose is required.

3.4.3 CRF layer

The difference between Bi-LSTM+CRF and purebred CRF is that in the Bi-LSTM+CRF model, Bi-LSTM replaces the potential function in purebred CRF, and the state transition probability of CRF and the decoding method are used to complement each other. The core code is showed below:

```
def crf_layer(self, logits):

  with tf.variable_scope("crf_loss"):
    trans = tf.get_variable(
        "transitions",
        shape=[self.num_labels, self.num_labels],
        log_likelihood, trans =
            tf.contrib.crf.crf_log_likelihood(
            inputs=logits,
            tag_indices=self.labels,
            transition_params=trans,
            sequence_lengths=self.lengths)
        return tf.reduce_mean(-log_likelihood), trans
```

3.4.4 Different on Sentiment Analysis

Compared with NER, the main change of SA code in the processing of Bi-LSTM network output and the new loss function.

```
logits = tf.matmul(output, output_weights,
    transpose_b=True)
logits = tf.nn.bias_add(logits, output_bias)
correctPred = tf.equal(
    tf.argmax(logits,1), tf.argmax(label_ids,1))
accuracy = tf.reduce_mean(
    tf.cast(correctPred, tf.float32))
loss = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits
    (logits=logits, labels=label_ids))
```

3.4.5 Overall calculation process

For a Named Entity Recognition or a Sentiment Analysis task, when we got a sequence, we first input the sequence into the BERT model. Then got a output of the hidden state, cause the multilingual BERT model has 768 hidden state, so we got a output with format [sequence_length * 768]. In this work we set the maximum sequence length as 128, so the out put will be 128 * 768. Then we send the out put sequence into Bi-LSTM layer and get the output. After that, we got two situations, one for NER task and another for SA task:

- For NER task: In entity recognition, we need to train each token to know the entity of itself, so we continue input the output of Bi-LSTM layer into the CRF layer , and then get the final prediction.
- For SA task: Because text classification only requires tags relative to the entire sentence, the results are directly expanded and then linearly changed. In fact, we can add another activation layer to the linear change, as long as you can ensure that the forward and backward propagation can proceed smoothly, the result is a result that can be converted into a label, not even something you don't know it. However, there are already some experiments proved that there is no improvement, and the previous network is sufficient.

Chapter 4

Experiments and Analysis

In this chapter, we will first introduce the basic environment of the experiment, train and experiment with the models and methods mentioned in the previous section. Both in 2 tasks and in 4 languages. Give the results of the experiment and conduct a detailed analysis and possible future research directions and work.

4.1 Experiment Environment

Here we will introduce the framework used to carry out our researches and some key packages which is of great help to our research implementation. The system environment and model options will also be listed.

4.1.1 Tensorflow

TensorFlow is a symbolic mathematics system based on data-flow programming, which is widely used in the programming of various machine learning algorithms. Its predecessor is Google's neural network algorithm library DistBelief. Tensorflow has a multi-level structure that can be deployed on various servers, PC terminals and web pages and supports GPU and TPU high-performance numerical computing. It is widely used in Google's internal product development and scientific research in various fields. It was released under the Apache License 2.0 in 2015[30]. Tensorflow, as a framework that is currently overly popular in deep learning, provides a very rich API, with three levels of APIs: stepwise, middle-level and high-level. The three-level API makes TensorFlow quite easy to use. We can use the middle and high-level API to quickly build the existing modules, and then use the low-level API to write some custom modules, which can greatly speed up our development and retain the flexibility of the code. As the lowest-level API, tensor is the core data unit of TensorFlow, which is essentially an array of arbitrary dimensions. The available tensor types include constants, variables, tensor placeholders, and sparse tensors. The calculation process of a tensor flowing from one end of the flow graph to the other end. It vividly describes the flow, transmission, analysis and processing mode of complex data structures in artificial neural networks.

4.1.2 System Environment

The working environment for performing the experiments are as follows:

- Jupyter Notebook: version 5.2.2
- Virtual environment: Google Colaboratory
- Python interpreter: version 3.6.9
- Network modeling and computation package library: Tensorflow, version 1.13.1, tensorflow-gpu, version 1.13.1
- GPU and GPU programming and acceleration: Tesla P100 and Tesla T4

Other development tools are also used such as perl, will be mentioned later in the following chapter.

4.1.3 Baseline Model Settings

The model Options of the baseline model are set as follows:

Embedding:

- Embedding dimension: 300
- Max sequence length: 128
- Shared embedding/Feature embedding/Positional encoding: disabled

Bi-LSTM Layer:

- Encoder type: standard RNN
- Gate type: Bi-LSTM
- Number of stacked layers: 2
- Hidden layer dimension: 128
- Learning rate: 0.001

- Dropout keep probability: 0.5
- Optimizer: Adam optimizer

Training Settings:

- Batch size: 32
- Epoch: 40

4.1.4 Bert Model Settings

The model Options of the Bert model are set as follows:

Pre-trained Model:

- Pre-trained Model: Bert multi_cased_L-12_H-768_A-12
- Epoch: 3
- Learning rate: 2e-5
- Batch size: 32
- Max sequence length: 128

In order to improve the efficiency of training, for Bert training, under the right conditions, the TPU provided by Google Colaboratory will be used. Normally will be the TPU v2. The specific models and performance are as follows[31]:

- 8 GiB of HBM for each TPU core
- One MXU for each TPU core
- Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod

4.2 Data Preparation

For NLP tasks, the selection of the data set has always been one of the core issues. Since two tasks need to be completed in this article, the selection of the data set is divided into two parts.

4.2.1 Named Entity Recognition Dataset

Sequence labeling is one of the basic problems we often encounter when solving NLP problems. In sequence labeling, we want to label each element of a sequence. Generally speaking, a sequence refers to a sentence, and an element refers to a word in the sentence. For example, the information extraction problem can be considered as a sequence labeling problem, such as extracting meeting time and location. Named entity recognition is a sub-task of the information extraction problem. It needs to locate and classify elements, such as the name of the person, the name of the organization, the location, the time, and the quality.

The easiest way to solve the joint labeling problem is to convert it into the original labeling problem. The standard practice is to use BIO tagging. BIO labeling: label each element as "B-X", "I-X" or "O". Among them, "B-X" means that the fragment in which this element is located is of type X and the element is at the beginning of the fragment, "I-X" means that the fragment in which this element is located is of type X and the element is in the middle of the fragment, and "O" means it is not one of any type. For example, if we represent X as a noun phrase (Noun Phrase, NP), then the three tags of BIO are:

- B-NP: the beginning of a noun phrase
- I-NP: the middle of the noun phrase
- O: Not a noun phrase

English Dataset

The shared task of CoNLL-2003 concerns language-independent named entity recognition. It concentrates on four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. The CoNLL-2003 shared task data files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag and the fourth the named entity tag[32][33].

The data is a collection of news wire articles from the Reuters Corpus. The annotation has been done by people of the University of Antwerp. The training data consists of 14041 sentences. The test set consists of 3453 sentences.

German Dataset

The GermEval 2014 NER Shared Task is an event that makes available CC-licensed German data with NER annotation with the goal of significantly advancing the state of the art in German NER and to push the field of NER towards nested representations of named entities[34]. The data set contains one token per line. It is similar to the CoNLL2003 format. Additionally the first column represents numbers per sentence and a commented line at the beginning of each sentence, representing the data source and data. The third columns contains the named entities in IOB format. The fourth column containing nested named entities is ignored in this task.

The GermEval 2014 data set contains text from German Wikipedia articles and onlines news texts. The training data consists of 24000 sentences. The test set consists of 5100 sentences. All in all the data set contains over 590000 tokens.

Chinese Dataset

Data in MSRA is collected from news domain and is used as shared task on SIGNAN backoff 2006. There are three types of named entities[35].

The MSRA, which the simplified character corpora provided by Microsoft Research Asia. The training data consists of 45000 sentences. The test set consists of 3442 sentences. All in all the data set contains over 2300000 tokens. The data set contains one token per line. It is also similar to the CoNLL2003 format.

Korean Dataset

We use the NAVER NER 2019 dataset for Korean dataset. The dataset is provided by a challenge from Naver and Changwon University. The data is provided by Naver, Korea's largest search provider, which consist of 81k training data and 9k test data.

4.2.2 Sentiment Analysis

Sentiment analysis has many classification. From word level and phrase level to sentence level, even into chapter level. Also depends on the task, it can be divide into the analysis based on comments, based on news, based on product reviews. Depends on the task that focuses on polarity (positive, negative, neutral), or to detect the emotions like angry, happy, sad. Even system for recognizing intent such as interested and not interested. The problems mentioned above can also be classified into fine-grained categories.

In this research, we mostly focus on the emotions expressed by user comments, so the granularity is at the sentence level. All data will be classified into sentiment, divided into positive and negative.

English Dataset

The english data is provide by the course of Hung-yi Lee, "Machine Learning and having it deep and structured". The data are tweets collected on twitter, each tweet will be marked as positive or negative. The dataset consists 200k twitters.

German Dataset

The GermEval 2017 dataset[36] contains the customer reviews about "Deutsche Bahn", which are from various social media and web sources. The dataset consists 2.2k reviews.

Chinese Dataset

The Chinese data set is the content of comments on Weibo, most of which are emotionally inclined. It contains 100k reviews, 50k for positive comments and 50k for negative.

Korean Dataset

The Korean dataset using Naver sentiment movie corpus. The corpus is a movie review dataset in the Korean language. In this corpus, all the reviews are shorter than 140 characters. There are in total 105k training data, 55k for positive and 50k for negative.

4.3 Baseline

In order to have a clearer understanding of the performance and improvement of the results of the experiment in different languages and different tasks, we need a baseline. This will be our evaluation standard for subsequent experiments. Therefore there are two baselines for two tasks in three different languages:

- Named Entity Recognition (NER): In order to control the variables, since we have connected the Bi-LSTM + CRF layer to the output of mBERT, our baseline building a simple character-based Bi-LSTM + CRF sequence labeling model. The NER task we use F1 score to evaluate the results.

- Sentiment Analysis (SA): It is slightly different from entity recognition, because text classification only requires tags relative to the entire sentence, so the result is directly expanded and then linearly changed. So we removed the crf layer of the NER baseline model, which also based on the vocabulary. The SA task we use accuracy to evaluate the results.

Language\Task	NER	SA
English	85.90	79.46
German	75.88	67.24
Chinese	88.77	86.24
Korean	85.20	80.24

Table 4.1: The baseline results in 2 tasks for 4 languages, which NER uses F1 score as evaluation and SA uses accuracy

4.4 Experiment on Named Entity Recognition

4.4.1 Results

For each language, we sample over 14000 sentences of each languages, most with Korean over 81000 sentences. All in IOB tagging with three entity labeled. Specific information is as follows:

- English: 14041 sentences
- German: 2.4k sentences
- Chinese: 4.5k sentences
- Korean: 8.1k sentences

After doing baseline on each language, we used the pre-trained M-BERT model provided by google[17], fine-tuning on each languages separately. Then in order to verify the cross-lingual transfer learning capability for low-resource language, we use the fine-tuned model of one language to evaluate on the other 3 languages. For example, we evaluate the German, Chinese and Korean test datasets with the fine-tuned English M-BERT model. With multiple times of adjusting the training parameters, we got a best setting for the model, showed in Table 4.2 and the results showed in Table 4.3. The horizontal represents the language of the tested data set. The vertical represents source language of the fine-tuned multilingual BERT.

Table 4.2: The best setting of the models

Parameter	Baseline setting	M-BERT setting
Best epoch	40	3
Learning rate	0.001	5e-5
Dropout rate	1.0	0.5
LSTM units	128	128

Fine-tuning\Eval	En	De	Zh	Kr
English	**94.82**	75.54	82.25	75.27
German	80.30	**87.45**	83.24	80.11
Chinese	77.71	71.69	**97.13**	82.03
Korean	78.44	72.63	85.42	**89.63**

Table 4.3: NER F1 scores on 4 languages

4.4.2 Analysis

Baseline vs mBERT

From the Figure 4.1 we can find the comparison between baseline and fine-tuned mBERT model, for each language, the performance of mBERT is significantly higher than the neural network model that only uses Bi-LSTM+CRF. This prove through fine-tuning the pre-trained BERT and combine the traditional neural network can get good improvement on NER tasks.

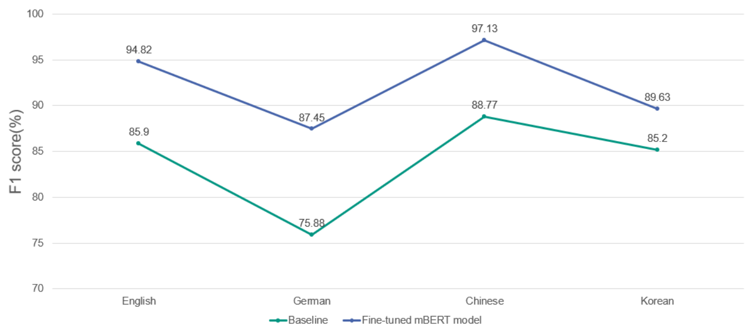


Figure 4.1: F1 result of Baseline and Fine-tuned BERT model in NER

Since the data volume of Chinese is significantly more than that of the English and German, we can also see that the F1 score of Chinese is also significantly higher than the other two languages, both in two models. Although the amount of data in Korean is the largest, it originally contained 14 tags, and the actual amount of annotated data required for the 4 types of tags in our experiment is also less than that of Chinese.

Cross-lingual transfer learning

Like showed in the Figure 4.2, each group means the language of evaluating dataset, the column represents evaluating on which model. Although have not been trained in the target language, all four transfer learning models can achieve good results on the evaluate of the other three languages tasks.

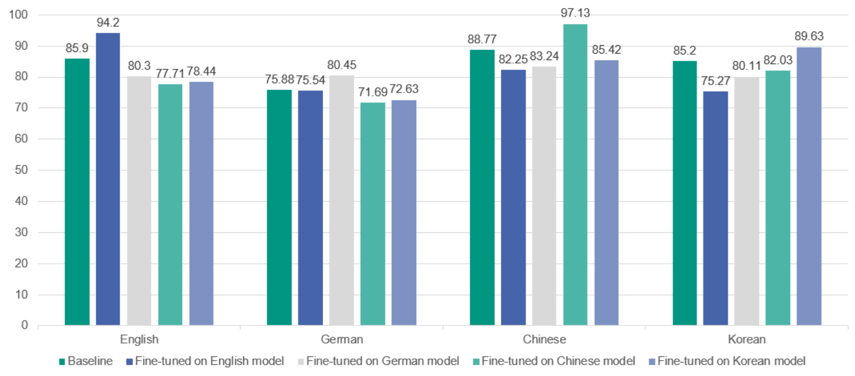


Figure 4.2: Cross-lingual transfer learning in NER task

For each language in NER task, We get the average F1 score of the three transfer learning model, and compare with the baseline in percentage in Table 4.5. It is not difficult to find that all the models can get a result more than 90% performance of baseline model, which is already a good result for low-resource NLP task.

In addition, we counted the number of labels in test data of each language, to explore the relation between the number of annotation and the results. We found that the two languages with the most abundant corpus resources, Chinese and Korean, have obviously more entity examples, and their average transfer learning results are better than the other two languages. However, although the Korean language has more corpus, the result is not higher than that of Chinese, nor is it much higher than the result of English. The reason may be that the source corpus of Korean contains too many interference options, so the real influence should be the ratio that number of entities and the total amount of data.

Furthermore, because these 4 languages all exist in the pre-training of multilingual BERT. In order to prove the versatility of the model to low-resource and even the zero-shot situation, we can attempt language tasks which the language does not exist in the pre-trained model in future work.

	En	De	Zh	Kr
Num. labels	5020	2371	6181	7824
Baseline	85.90	75.88	88.77	85.20
Transfer Learning Avg.	78.82	73.29	83.64	79.14
Comparison(%)	91.8	96.59	94.22	92.88

Table 4.4: Transfer learning F1 score comparison in percentage

Effect of language similarity

In addition, we can know that English and German have similar forms which both use letters, Chinese and Korean have similar forms and both use characters. Therefore, we compare the trends of the F1 score of different languages on different pre-trained models. As shown in the figure below, the results verify our conjecture. The English and German datasets have obviously achieved better results on each other's pre-trained models. Chinese and Korean model have the same effect.

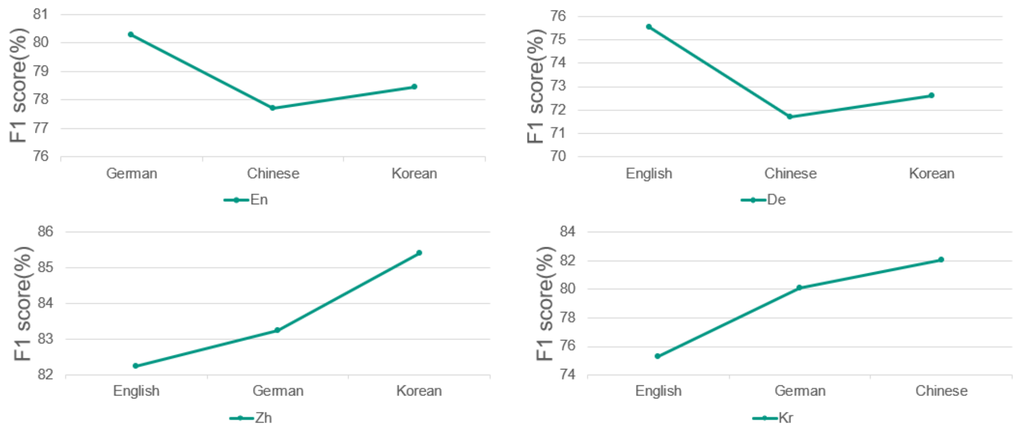


Figure 4.3: F1 result of Baseline and Fine-tuned BERT model in NER

4.5 Experiment on Sentiment Analysis

4.5.1 Results

Different from the entity recognition, our sentiment analysis deal with sentence or small paragraph with emotion comment in social media like twitter and weibo. Specific information is as follows:

- English: 200k twitters
- German: 2.2k social media comments
- Chinese: 100k weibos
- Korean: 105k Naver movie comments

Parameter	Baseline setting	M-BERT setting
Best epoch	40	3
Learning rate	0.001	5e-5
Dropout rate	1.0	0.5
LSTM units	128	128

Table 4.5: The best setting of the models

Like training on NER tasks, after doing baseline on each language, we used the pre-trained M-BERT model provided by google[17], fine-tuning on each languages separately. Then doing exactly the same thing as NER task. Evaluate the low-resource language dataset using the fine-tuned multilingual BERT model. Unlike twitters or weibos in English, Chinese and Korean datasets only composed of one or two sentences. The German data usually is a small paragraph, but considering the computing power of the machine, we set the maximum sentence length to 128. With multiple times of adjusting the training parameters, we got a best setting for the model, showed in Table 4.5 and the results in Table 4.6:

4.5.2 Analysis

Baseline vs mBERT

Although the input in the sentiment analysis task is very different from NER, the results are very similar. From the Figure 4.4 we can find the comparison between baseline and fine-tuned mBERT model, for each language, the performance of mBERT

Fine-tuning\Eval	English	German	Chinese	Korean
English	**83.01**	58.46	69.70	58.18
German	66.89	**88.93**	62.85	55.61
Chinese	60.13	55.20	**96.92**	62.17
Korean	63.25	50.12	71.24	**86.47**

Table 4.6: Accuracy results on 4 languages in Sentiment Analysis

is significantly higher than the model that the traditional neural network using Bi-LSTM+CRF. Since the German dataset is composed of many sentences and long paragraphs, we found that it is obvious that traditional neural network methods cannot achieve good results. In contrast, the method using BERT has a very good performance, which got the biggest improvement in 4 languages. This also reminds us that the preparation of the data set is important.

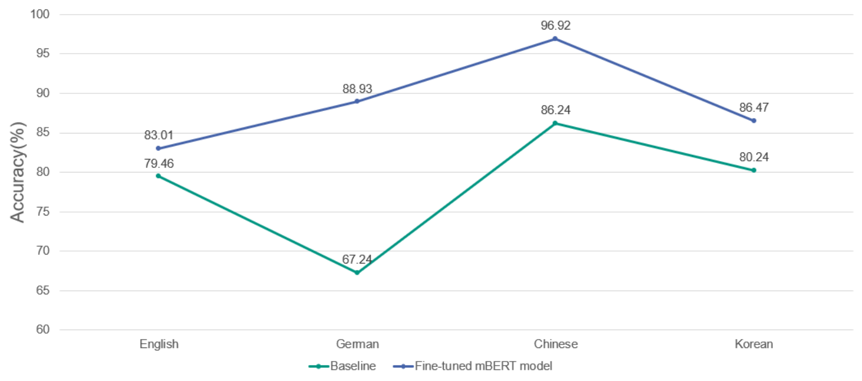


Figure 4.4: Accuracy of Baseline and Fine-tuned BERT model in Sentiment Analysis

Cross-lingual transfer learning

In the same representation like NER, we set the cross-lingual comparison in Figure 4.5. The group means the language of evaluating dataset, the column represents evaluating on which model. We can see the result not good as the named entity recognition.

We calculate the average accuracy of three transfer learning model of each language, comparing with the baseline in Table 4.7, except due to the low performance of German, both model can't get over 80% performance of baseline model.

After review the structure of the BERT transfer learning. The reason may be that

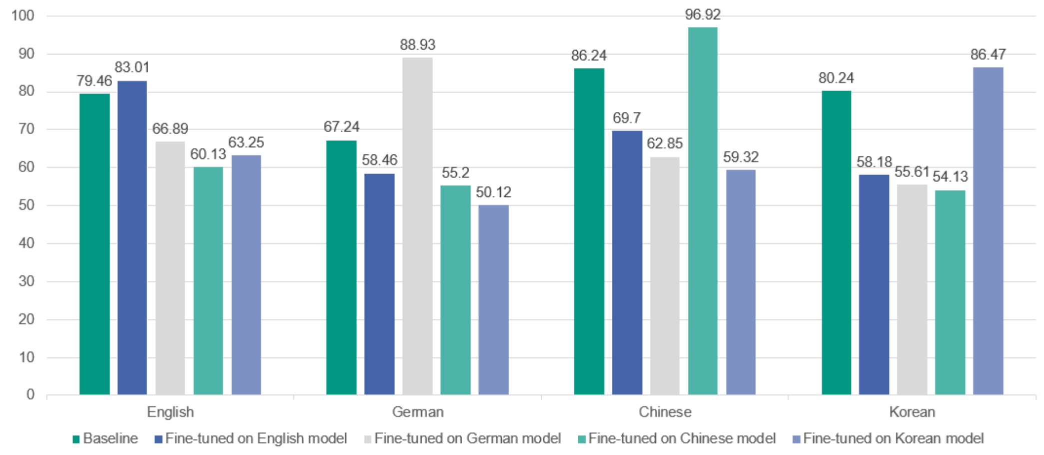


Figure 4.5: Cross-lingual transfer learning in Sentiment Analysis task

BERT analyzes WordPiece and puts it into the shared embedding space. Even maybe the word embedding has the same or similar meaning and got the close distance in shared embedding space, the words still can express the different emotion in different language. Whats more, not like NER, the sentiment of one or two words normally can't decide the sentiment of the whole sentence or a small paragraph. Nevertheless, for low-resource languages, we still think that the results that can achieve 70% baseline performance are still acceptable.

	En	De	Zh	Kr
Baseline	79.46	67.24	86.24	80.24
Transfer Learning Avg.	63.42	54.59	67.93	58.65
Comparison(%)	79.81	81.19	78.77	73.09

Table 4.7: Transfer learning Accuracy comparison in percentage

Effect of language similarity

In view of the enlightenment of the NER results, we also compared the impact of the similarity of different languages. We were surprised to find that despite the unsatisfactory performance, the results still met the conclusion similar to NER task. We can find, the English and German datasets still have achieve better results on each other's pre-trained model.

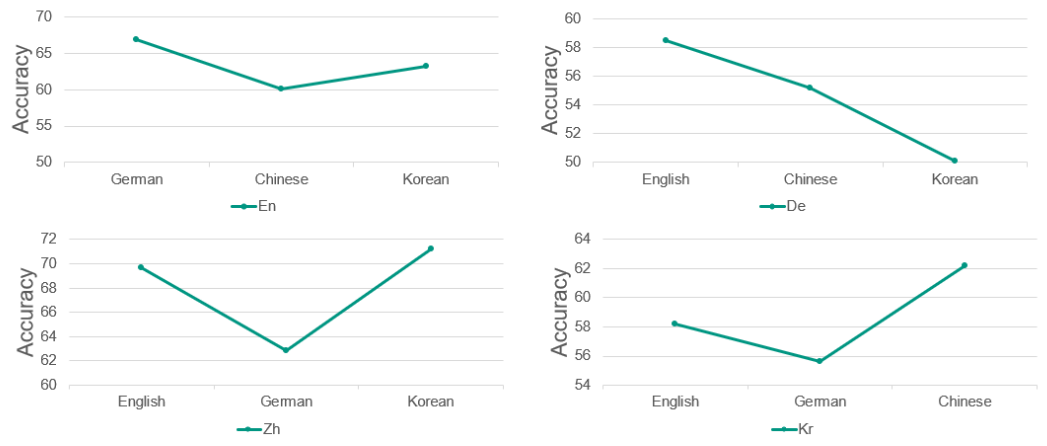


Figure 4.6: F1 result of Baseline and Fine-tuned BERT model in NER

4.6 Future Work

Through the whole experiment and thinking during the process, as well as the analysis of the experimental results. And some things don't get enough time and resource to do. We think there are some things that can be studied in depth and some parts that can be explored:

- For a traditional sequence labeling problem, NER has good cross-lingual transfer learning performance in the four languages mentioned in the article, and we can extend this to other tasks, like part-of-speech tagging, question answering etc.
- The language tested in the thesis are all included in the pre-trained training data, we can explore for the zero-shot case, which the data not included in the pre-trained model, how will be the performance of the model.
- We did not achieve good enough results in sentiment analysis task. Because I think the main reason is that the tokenization method on the input side cannot handle the tasks which input in units of sentence. In the future attempt we can use another fine-tuning method mentioned in the thesis, change some layer of the model to change the input method.
- Since it has achieved very good results on the two tasks, we can explore how multilingual BERT achieves such a good cross-lingual transfer learning effect, and explore whether it has the same effect in all languages.

- Further explore the influence of language similarity on the results of cross-lingual transfer learning. Limited by the scale, only 4 languages were tested in this work. We can adapt the same system in further languages, and quantitatively evaluate the similarity of languages.

Chapter 5

Conclusion

In this work, we propose a cross-lingual transfer learning model to solve the low-resource NLP tasks. With doing much experiments on 2 tasks with multiple languages, we proved our model can achieve good results on these tasks. I hope to conduct some analysis on the above experiments and draw some conclusions of my own and ideas for future experiments.

After the above experiment, we have proved that through the fine-tuning the multilingual BERT on a large corpus source language NLP task, we can transfer the model on a same task with low-resource target language. Although compare the two experiments we conducted, sentiment analysis got a much worse results than name entity recognition. But both model can be seen as well done on low-resource language tasks. Combine two tasks which during the experiments, there are several points which I think play big roles in this research:

- The proportion of labels in the data set: As we all know, data sets in deep learning is one of the most important factor. In our transfer learning task, for example in NER task, although the Korean corpus has the most labels and tokens, the result still lower than that in Chinese. Same in sentiment analysis, the German corpus consist of small paragraph and many sentences, so the results is obviously low than others.
- The form of the languages: For both training language and low-resource target language, the language form is very important. The closer the form of the language token, the better the performance of cross-lingual processing. For example, in the named entity recognition task, the cross-lingual performance between English and German is significantly better than the performance between Chinese with other two languages. Cause the English and German both using alphabet as the input token and Chinese uses character.

- Language in pre-trained model: Though we the cross-lingual transfer learning based on fine-tuned multilingual BERT got good results in low-resource task, but the tested low-resource language still consist in the large monolingual corpus in pre-trained model. We think this is also why the transfer learning can achieve such a good result.

The factors mentioned above have a very important impact on performance in the cross-lingual transfer learning of low-resource NLP task. We can obviously see the progress of the new model on low-resource language.

We have only conducted experiments on two tasks of natural language processing, according to this idea, we can adapt the model to other NLP tasks in the future. In addition, for the processing of BERT output, we use the more general Bi-LSTM and CRF models, maybe we are able to adopt other methods for processing to explore whether we can obtain better results. With the improvement of above mentioned factors and model structure, I think the cross-lingual transfer learning on low-resource NLP tasks can achieve a better result than the traditional method which needs large corpus.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[3] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] When to use recurrent neural networks. https://iq.opengenus.org/when-to-use-recurrent-neural-networks-rnn/. Accessed Oct 7, 2020.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[10] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[11] *Conditional Random Fields Explained.* https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776.

[12] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[13] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[16] Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631, 2019.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[19] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[20] Thanh-Le Ha, Jan Niehues, and Alexander Waibel. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*, 2016.

[21] Tsung-Yuan Hsu, Chi-Liang Liu, and Hung-yi Lee. Zero-shot reading comprehension by cross-lingual transfer learning with multi-lingual language representation model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on*

Natural Language Processing (EMNLP-IJCNLP), pages 5933–5940, Hong Kong, China, November 2019. Association for Computational Linguistics.

[22] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual bert? *CoRR*, abs/1906.01502, 2019.

[23] Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. Cross-lingual BERT transformation for zero-shot dependency parsing. *CoRR*, abs/1909.06775, 2019.

[24] Shijie Wu and Mark Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*, 2019.

[25] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR, 2020.

[26] Chi-Liang Liu, Tsung-Yuan Hsu, Yung-Sung Chuang, and Hung-Yi Lee. A study of cross-lingual ability and language-specific information in multilingual bert. *arXiv preprint arXiv:2004.09205*, 2020.

[27] Jindřich Libovickỳ, Rudolf Rosa, and Alexander Fraser. On the language neutrality of pre-trained multilingual representations. *arXiv preprint arXiv:2004.05160*, 2020.

[28] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[29] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics.

[30] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[31] Google cloud. Google cloud official site. https://cloud.google.com/tpu/docs/system-architecture.

[32] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[33] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[34] Germeval2014 info. https://sites.google.com/site/germeval2014ner/home?authuser=0.

[35] Gina-Anne Levow. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia, July 2006. Association for Computational Linguistics.

[36] *Germeval2017 dataset*. https://sites.google.com/view/germeval2017-absa/home?authuser=0.