Named Entity Recognition with Deep Learning

Haobin Yu

A thesis submitted to Auckland University of Technology
In partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2019

School of Engineering, Computer and Mathematical Sciences

Abstract

Named entity recognition is a significant subtask of natural language processing. Traditional methods require large numbers of feature engineering and handcrafted additional dictionaries to achieve high performance.

Nowadays, deep learning is widely applied for image and natural language processing domains. Through the deep neural network, features are automatically extracted from the training data, and this avoids most feature engineering. Some deep neural network based methods are also used for the modelling sequence labelled problem. Throughout these methods, the performance of named entity recognition can be improved.

In this thesis, we have mainly used the bidirectional LSTM model based on deep learning as an architecture, word embedding and Convolutional Neural Network (CNN) as a word-level and char-level feature extractor. The conditional random field (CRF) layer is used to output the predicted labels. We used the CoNLL 2003 dataset and the Broad Twitter Corpus dataset training and testing the Bi-LSTM-CNN-CRF model respectively and got satisfactory results. In order to evaluate the Bi-LSTM-CNN-CRF model, we compared it with three popular machine learning methods. The three popular machine learning methods include the Support Vector Machine (SVM), the Hidden Markov Model (HMM), and the Conditional Random Field (CRF). The results of the evaluation show that the Bi-LSTM-CNN-CRF model with only given labelled text and pre-training word embedding surpasses the three machine learning methods that employ handcrafted feature engineering.

Keywords: Named Entity Recognition, Deep Learning, Deep Neural Network, Convolutional Neural Network (CNN), Bi-LSTM, Support Vector Machine (SVM), Hidden Markov Model (HMM), Conditional Random Field (CRF), Word Embedding.

Contents

List of Figures

List of Tables

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person except that which appears in the citations and acknowledgements. Nor does it contain material which to a substantial extent I have submitted for the qualification for any other degree of another university or other institution of higher learning.

Signature: Date:09/07/2019

Acknowledgment

From the beginning of this project to the end I have received much help from many people. First of all, I would like to express my special thanks of gratitude to my supervisor Parma Nand who gave me the golden opportunity to do this wonderful project, which also helped me in doing a lot of research and I came to know about so many new things. I am really thankful to him.

Finally, I would like to thank my classmates, parents, and friends who have been with me for one and half years, thank you for all the helpful suggestions and opinions they have given me, and thank them for helping me find the materials during the writing process, so that I can successfully complete the thesis writing. Thanks for their support and help in my life!

Haobin Yu

Auckland, New Zealand

8 July 2019

Chapter 1 Introduction

This chapter is composed of five parts. The first part introduces the background of named entity recognition briefly and the motivation for this project. The second part presents the questions this thesis attempts to explain. The results that are expected to be achieved will be introduced in the fourth part. The structure of this thesis is introduced in the last part.

1.1 Background and Motivation

Nowadays, there are many sequence modelling methods that are applied to natural language processing (NLP), information extraction, machine translation, speech recognition, and so forth. A named entity generally refers to an entity with a specific meaning or strong reference in the text, usually including the person's name, place name, organization name, date and time, proper noun, and the like. The named entity recognition (NER) system extracts the above entities from the unstructured input text and can identify more of the categories of entities according to the business needs, such as, product name, company name, price, and so on. Therefore, the concept of an entity can be very broad, as long as it is a particular piece in the text that the business needs, it can be called an entity. The NER is based on the task of NLP and it has been used for a wide range of applications. Therefore, how to apply sequence modelling technology to improve the accuracy of NER and then further improve the downstream NLP tasks is essential.

In the past, rule-based and dictionary-based methods are used to recognise the named entities. They rely on handcraft rules for human's language grammar and entity libraries. The disadvantage of the rule or dictionary-based approach is that it is too costly, and there are problems such as the long period of system construction, poor portability, and the need to establish different domain knowledge to enhance system identification capabilities (Nadkarni, Ohno-Machado & Chapman, 2019).

Recently, based on machine learning, people start through supervised learning building statistical models to calculate the probability distribution to classify and recognise entity tags. Researchers use generative models instead of discriminative models to improve the performance of models in supervisor learning. Researchers also extend the models to unsupervised learning to decrease handcraft feature extraction. We use one discriminative model, the support vector machine (SVM), and two generative models, the Hidden Markov Model (HMM) and the Conditional Random Field (CRF).

Deep learning has avoided manual feature extraction, through unsupervised learning or semi-supervised learning it extracts and learns features to build the model. The bidirectional long short-term memory network (Bi-LSTM) (Graves & Schmidhuber, 2005) is widely applied for the sequence label problem with long term dependence. It can improve the accuracy of the overall training. In this thesis, we use word embedding and char-level embedding as the feature extractor, and combine it with external pre-training vectors and feed them into the framework of Bi-LSTM-CRF model (Ma & Hovy, 2016).

In the process of NER, we first need to prepare the data for different models, because our models require a different data format. Next, we need to spend time tuning the parameters of the neural network. Our neural network uses different optimization methods. Parameter tuning contributes to the performance of the model.

1.2 Research Question

The introduction summarises the basic knowledge of this thesis, which is about classifying and identifying named entities. A few research questions have been set forth to further understand the processing of NER. The research question is as follows:

Does Deep Learning improve the performance of Name Entity Recognition?

The following sub-question will help us better solve the major question.

What is Name Entity Recognition?
Which algorithms is the best one apply for Name Entity Recognition?
What is the process of Name Entity Recognition system?

The main research question is about NER, to achieve the goal we need to find the algorithms and the related techniques that apply to our project and we need to understand the specific processing of the NER system.

1.3 Contributions

The main argument of this thesis involves building four models for NER and comparing their results. According to the procedure that is proposed in this thesis a NER experiment is performed step by step. In this thesis, we introduced the following steps:

- Data collection and pre-processing,
- Build a support vector machine model,
- Build a Hidden Markov model,
- Build a Conditional random field model,
- Char-level feature extraction through Convolutional neural network,
- Building Bi-LSTM-CRF model,
- Named entity classification and recognition by four models,
- Results analysis and comparison.

The details of the methods in terms of the related algorithms that are used in our experiment are introduced in Chapter 2. The particulars of the methods that are used in our experiments are introduced in detail in Chapter 4.

The contribution of our research project is:

- NER based on machine learning,
- NER based on deep learning,
- How to use the CNN model as char-level word embedding,

- Comparison of different of four models,
- Parameter tuning of Bi-LSTM in details.

Our work is based on machine learning and deep learning. Our research can meet the needs of NER.

1.4 Objective of this Thesis

Firstly, the processing of the NER system is divided into two parts: training model and prediction. We will introduce this in this thesis in detail.
In order to recognise named entity in different articles, including informal tweets, we need to consider further feature extraction from diverse corpus to build the models.
For this research project, we have used four methods to build the models and we have compared these results to find the optimal methods and then we discuss the limitations of the models. In addition, in the thesis, we will compare the four methods in detail together with our experimental results.

1.5 Section Introduction

This thesis is comprised of five Chapters, and each section has the following content:
In Chapter 2, we will introduce the previous work in detail. We focus on the literature that has studied NER and popular sequence models. The first part is the literature review of the concept of NLP, and the important models and their applications. Then, we introduce NER and methods implementation. Finally, we review the different models for feature extraction, classification, and predictions based on machine learning and deep learning.
In Chapter 3, we introduce the particular methods that we use. We introduce four models and their theories. We also present correlation algorithms for implementing a complete NER system. In addition, we also discuss the basic workflow of the NER system.
In Chapter 4, we introduce the implementation of NER. We also include data collection and pre-processing. The results of our experiments will show as diagrams in this Chapter. In addition, we compared four methods under a different corpus. We describe the results in detail, and we discuss the limitations of this project.
The comparison of the principle of four methods is introduced in Chapter 5. We also introduce the parameter tuning of the Bi-LSTM model in detail.
Chapter 6 presents the conclusion and discusses future work.

Chapter 2 Literature Review

In this chapter, we analyse the natural language processing and named entity recognition via a review of the past research. Through the literature review, we can understand the theoretical knowledge and the application methods of named entity recognition. This chapter also introduces the methods that mainly apply for named entity recognition.

2.1 Introduction

With the continuous advancement of the research and development of NLP, the neural network instead of traditional machine learning has become the mainstream in term of the research (LeCun, Bengio & Hinton, 2015). Based on several new optimization methods the neural network and word representation methods improve the feature selection and overall performance of many NLP tasks.

Through a review of the past research this paper can briefly understand the development of NLP. It has helped us answer the sub-research questions and choose the methods to implement the experiment. And then we introduced NER and the development of the methods applied for NER.

The second part of this Chapter mainly introduces each method used in this project. After a review of methods applied for NER, from machine learning based methods, we selected the SVM, the HMM, and the CRF as our implementation methods. In addition, we also chose to use the Bi-LSTM method as the other method for this project.

After reviewing the four methods that we chose for this project, we have the basic idea for the implementation of the workflow for the whole project. The first step is to start the data collection and pre-process the data. The next step is using the methods and algorithms to build the models. In order to identify the named entity in the articles, we need to select the features to support the model.

2.2 Overview of Natural Language Processing

Natural language processing (NLP) is a cross-cutting field of artificial intelligence, computer science, and information engineering, involving knowledge of statistics, linguistics, and so forth (Zagal, Tomuro, & Shepitsen, 2012). With respect to NLP, the computer accepts the input of the user's natural language, and it simulates human understanding of natural language through a series of human-defined algorithms to process and calculate and then return the results of the user's expectations (Daelemans, & Van den Bosch, 2005). The purpose of NLP is to use computers instead of a person or a team of people to process large-scale natural language information.

NLP has lots of applications, such as, machine translation, sentiment analysis, chat robot, and intelligent customer service. The basic technologies that are involved include word segmentation, part-of-speech tagging, NER and named entity disambiguation, and so forth.

Before 2000, the mainstream of NLP is statistical methods. Researchers focus on extracting more features from articles to improve the performance of the model, such as, the gazetteer, and so on.

In 2001, a feed-forward neural network (Bengio, Ducharme, Vincent & Jauvin, 2003) combined neural networks and language models, the network used neural networks to obtain word embedding matrices, which is the practical basis for all subsequent word embedding techniques. The possibility of a neural network language model has also been demonstrated.

The CRF has been proposed to handle labelling sequence data (Lafferty, McCallum & Pereira, 2001). This method avoids the strong independence assumption of the Markov chain and related model. It is even often added to the neural network model to correct the output sequence.

In 2003, the latent Dirichlet allocation (Blei, Ng & Jordan, 2003) model was proposed so that the probability map model, is widely applied for topic modelling. There are many variants of the topic model, such as, the supervised Label LDA (Ramage, Hall, Nallapati, & Manning, 2009), the PLDA (Wang, Bai, Stanton, Chen & Chang, 2009), and so forth.

Statistical methods focus on the distribution of the data itself. The main aspect of the statistical method is involved with how to design more and better feature patterns from the distribution of texts.

Mikolov et al. (Mikolov, Chen, Corrado & Dean, 2013), proposed word2vector which is an important technology that has to do with the neural network.

With the development of computing power, the neural network can be deeper and deeper, and the previously restricted neural network is no longer in the theoretical stage. After it has been proved in the image domain, the deep neural network is applied for NLP.

Due to the increase in computing power the calculation of neural networks are no longer limited. Although the neural network is a black box, it saves a lot of design features.
In 2014, sequence to sequence learning (Sutskever, Vinyals & Le, 2014) was proposed which is a framework that can map one sequence to another sequence through the neural network. Machine translation has gained huge improved performance by using this framework. Google used the sequence to sequence framework with the attention model to replace its monolithic phrase-based machine translation models (Wu et al., 2016). Under the framework, the encoder and the decoder can be different models, thus this framework is also applied for many other NLP tasks, such as, generating a caption based on an image (Vinyals, Toshev, Bengio & Erhan, 2015), generating text through structured data (Lebret, Grangier & Auli, 2016), and generating natural descriptions from source code changes (Loyola, Marrese-Taylor& Matsuo, 2017).
Attention was proposed in 2015. It is the crucial function that causes neural machine translation to be better than the classic phrase-based machine translation system (Bahdanau, Cho & Bengio, 2014). In addition, attention is widely applicable, it is used for any task that requests making decisions through input data potentially. It is not only used for NLP field it is also applied for image captioning (Xu et al., 2015) Multiple layers of self-attention are also used for the Transformer architecture (Vaswani et al., 2017).
Pre-training word embedding is context-agnostic, it is only used for the first layer initialization, it can be used as features add into the neural network model (Ramachandran, Liu & Le, 2016) (Peters et al., 2018). It was shown to be beneficial across a various range of tasks, such as, it was fine-tuned on the target task data (Ramachandran, Liu & Le, 2016). At the end of 2018, BERT (Devlin, Chang, Lee & Toutanova, 2018) was proposed to deal with 11 NLP missions. It created state-of-the-art models for a wide range of tasks. Neural networks can automatically extract the features from the data, and in terms of people it can separate their complex features, and focus more on the innovation of the model algorithm itself and the breakthrough of the theory.

2.3 Named Entity Recognition

Named entity recognition (NER) is a subtask of NLP, the purpose is identifying named entity mentioned in articles into pre-defined categories, such as, a person's name, organizations, locations, times, date, currency, and percentage. From the whole process of text analysis, NER belongs to the field of unknown word recognition. It is an indispensable component of various NLP tasks, such as information retrieval, machine translation and so on. On the other hand, with the development of artificial intelligence technology, more and more application scenarios based on NLP, such as, machine translation, chat robot, and intelligent customer service will use this technique. This task must first be overcome in most artificial intelligence research related to NLP. With the

rapid development of mobile Internet and information technology, the number of texts generated by news, commentary and social media has exploded. Manually processing large amounts of textual information becomes increasingly difficult. Therefore, based on techniques of NLP, for example, information retrieval and intelligent customer service will play a more significant role. However, compared with a huge amount of text data, only a few amounts of data were available for supervised training, so it's difficult to generalise from these few samples of data. Thus, nowadays, carefully constructed handcraft engineered features and specialized language knowledge resources are used to solve named the entity recognition problem (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016).

2.3.1 Textual Category or Domain

In the early researches, the corpus usually collected from news stories, such as CoNLL 2003 shared task and so forth (Sang & De Meulder, 2003). Many other fields of text have been ignored, such as, scientific articles or research studies, informal text, sports, business, and so forth (Nadeau, 2007).

Recently, the rapid development of so many disciplines, numerous new publications appear daily in academic journals researchers are challenged to stay on the cutting edge of research. In order to advance many fields of research and avoid repetition researchers gain latest research development by reading published information. Therefore, extracting and searching for information from a huge number of databases has become very important. In order to achieve this goal, the very fundamental pre-processing step is to identify the named entity mentioned in the documents. After this step, the downstream NLP tasks, such as, information search and information extraction can improve the efficiency in terms of accessing the information (Hemati & Mehler, 2019). However, for fields, such as, biological, chemical, and biomedical research, many chemical and drug names are difficult to recognise.

The most difficulty in biological NER is the lack of standardization of the names. There are several names and abbreviations for the same gene or protein. Some abbreviation of biological terms is the same as the common English word. Resolving the ambiguity in gene or protein names is a significant problem (Jensen, Saric, & Bork, 2006).

In 2002, a rules-based named entity recogniser was designed based on the characteristics of biological data, which is used to identify six types of biological named entities by Narayanaswamy et al., (Narayanaswamy, Ravikumar, & Vijay-Shanker, 2002). In 2004, Kim et al., published bio-entity recognition shared task at JNLPBA based on GENIA version 3 named entity corpus (Kim, Ohta, Tsuruoka, Tateisi, & Collier, 2004). This paper introduces many machine learning methods. Chang et al., and Finkel et al., combined machine learning methods and dictionary matching or rules-based system (Chang,

Schütze, & Altman, 2004) (Finkel et al., 2005). Si et al., Smith et al., and Zhu et al., proved that combining multiple machine learning algorithms can achieve a higher performance of recognition (Si, Kanungo, & Huang, 2005) (Smith et al., 2008) (Zhu, & Shen, 2012).

The same problem happens in clinical and public health research. In order to transform massive health datasets to valuable knowledge, professional terminology from electronic health records, clinical handover notes, discharge summaries, and the records of clinical nursing handover need to be recognised. Patients and laypersons usually find that it is quite difficult to understand discharge summaries, handover notes, and other eHealth documents. Even researchers also have issues in understanding the jargon of other professional groups (Luu, Phan, Davey, & Chetty, 2018).

Biomedical NER uses the same methods with biological NER system, such as, the CRFs based BANNER system (Leaman, & Gonzalez, 2008), the two-phase model based on the SVM (Lee, Hwang, Kim, & Rim, 2004), the HMM-based model (Zhang, Shen, Zhou, Su & Tan, 2004) and so forth.

In 2015, Li et al., used an extended recurrent neural network for biomedical NER (Li, Jin, Jiang, Song, & Huang, 2015). Then deep learning was used in this domain. The Bi-LSTM and other neural network methods and neural network based word representation features improve the biomedical NER (Wu, Xu, Jiang, Zhang, & Xu, 2015) (Jagannatha, & Yu, 2016) (Habibi, Weber, Neves, Wiegandt & Leser, 2017).

In addition to the standard NER tools that get bad performance in professional fields, the performance of standard NER tools also has been seriously degraded in noisy, informal and unstructured data (Ritter, Clark & Etzioni, 2011) (Derczynski et al., 2015). Twitter data includes lots of misspellings, unreliable capitalizations, and grammatical errors. Therefore, traditional methods do not work well in tweets (Li, et al., 2012). To solve this problem, Alan Ritter et al., re-built the NLP pipeline, they focus on the performance enhanced part of speech tagging and chunking in Twitter data and then used them to improve the performance of NER (Ritter, Clark & Etzioni, 2011). In 2015, Baldwin et al., normalized the Twitter data and used the machine learning methods to identify the named entities (Baldwin et al., 2015). Godin et al., used word embedding and feedforward neural networks instead of traditional features selection and achieved higher performance (Godin, Vandersmissen, De Neve, & Van de Walle, 2015). In 2016, Bidirectional LSTM was used for Twitter NER (Limsopatham, & Collier, 2016). Through a combination of multiple word representations methods as the word representation layer in a neural network model have proven to be effective (Dugas, & Nichols, 2016) (Lin, Xu, Luo, & Zhu, 2017).

2.3.2 Languages

While English probably is the most researched language in NER, there are many other languages studied by NER (Nadeau, 2007). Since ConLL 2002 and ConLL 2003 shared task (Sang & De Meulder, 2003), Spanish, Dutch, and German were researched in NER. Japanese (Isozaki, 2001) and Chinese (Zhang, Yu, Xiong, & Liu, 2003) study also started very early. Other European languages, such as, Italian (Bonadiman, Severyn, & Moschitti, 2015), French (Palmer, & Day, 1997), Greek (Petasis et al., 2001), Spanish (Cotik, Rodríguez, & Vivaldi, 2018), and Turkish (Yeniterzi, Tür, & Oflazer, 2018) and so forth have also been researched. Asian languages, for example, Thai (Aroonmanakun, Nupairoj, Muangsin, & Choemprayong, 2018), Indonesian (Wibawa, & Purwarianti, 2016), Vietnamese (Dong, & Nguyen, 2018), Malay (Salleh, Asmai, Basiron, & Ahmad, 2017), and Korean (Choi, & Cha, 2016) and so on were researched by many researchers. Russian (Mozharova & Loukachevitch, 2016) and Arabic (Zirikly & Diab, 2015) were studied by many researchers as well.

2.3.3 Entity Type

The NER method relies heavily on training data, while early databases mostly came from news articles, such as, the famous CoNLL 2003 corpus. CoNLL 2003 divides the data into four major categories, people, locations, companies, and miscellaneous (Sang & De Meulder, 2003). Current NER tools classify entities into seven types (person, location, organization, amount, time, production, and function) (Galibert, Rosset, Grouin, Zweigenbaum, & Quintard, 2012). However, the fine-grained categorization of entities is necessary because these simple categories are no longer sufficient to meet the complex demands (Liu & Birnbaum, 2007). It lacks annotations for fine-grained categories, such as, for movie titles, singers, and so forth.

In 2001, Fleischman used machine learning methods to classify location types into fine-grained categories and meet or exceed human standards (Fleischman, 2001). Then Fleischman and Hovy designed a method to classify the person type into eight sub-fine-grained categories through context, WordNet and topic signatures (Fleischman, & Hovy, 2002). In 2005, Lee et al., combined external linguistic resources and the bootstrapping algorithm to develop the improved performance of the NER with fine-grained geographic classes (Lee & Lee, 2005). Since Srihari combined the Maximum Entropy Model (MaxEnt), the HMM and handcrafted grammatical rules to recognized three main types and 21 sub-types (Srihari, 2000), after that, machine learning methods were widely used in fine-grained NER. In 2006, Lee et al. (Lee et al., 2006), used the CRF to identify 147 subcategories in the question-answering system corpus. Ling et al., used the CRF model

to segment training data and then used the perceptron algorithm to deal with 112 categories (Ling & Weld, 2012). In 2015 Yogatama et al., researched the fine-grained NER using word embedding as word representation (Yogatama, Gillick, & Lazic, 2015). Mai et al., compared machine learning based methods and neural network-based methods and found the neural network-based methods work well with fine-grained NER (Mai et al., 2018).

2.3.4 Techniques and Algorithms to Solve the NER Problem

2.3.4.1 Rule-based and Dictionary-based Methods

Rule-based and dictionary-based methods are the earliest methods used in NER (Ji et al., 2019). Simulating human language was the mainstream of NLP at this time, such as, context-free grammar (CFG) was proposed by Chomsky (Nadkarni, Ohno-Machado, & Chapman, 2011). They rely on handcrafted rules, use named entity libraries, and assign weights to each rule. When a rule conflict is encountered, the rule with the highest weight is selected to determine the type of the named entity. In general, the performance of rule-based methods is better than statistical-based methods when extracted rules can more accurately reflect linguistic phenomena. However, these rules often depend on the specific language, domain, and text style (Ji et al., 2019). The portability of the system is not satisfactory. Linguistic experts have to rewrite the rules apply for different languages and domain. The compilation of rules into a dictionary is time-consuming. Another problem of the rule-based method is difficult to cover all linguistic phenomena.

2.3.4.2 Machine Learning based Methods

NER is a sequence labelling problem. It is based on the input sequence. The data is sequentially labelled. Nowadays, the main technical methods for NER include the machine learning based method, the neural network method, and the mixed model.
The methods that are based on machine learning include the HMM, the SVM, and the CRF, and so on. They all need a large annotated corpus that stores various features.
In 2002, according to Zhou and Su, they entered simple deterministic internal features of the words, internal semantic features of important triggers, internal gazetteer features, and external macro context features into their HMM model. They got high performance in newswire based data (MUC-6 and MUC7) (Zhou & Su, 2002).
In 2004, in regard to the HMM-based model that is used in biomedical NER, Zhang et al., used more features that were based on useful for biomedical entities, such as, head noun triggers features and special verb triggers features (Zhang, Shen, Zhou, Su, & Tan,

2004).

In 2007, Ponomareva et al., focused on only through parts-of-speech tags to solve the non-uniform distribution of biomedical entity categories and they achieved good results (Ponomareva, Pla, Molina, & Rosso, 2007).

In 2004, Zhao researched unlabelled biomedical NER, he used the word similarity-based smoothing algorithm and it improved the overall performance (Zhao, 2004).

In 2003, character-level models proved to be better than word-level models. Using the same HMM model, 30% of the error reduction from the word model was achieved compared to a character model (Klein, Smarr, Nguyen, & Manning, 2003).

In the NER task, chunking is very important in terms of pre-processing for data. BIO and BILOU are two of the most popular text segment schemes. BIO represents the Beginning, the Inside and the Outside of the named entity. BILOU can identify the Beginning, the Inside and the Last tokens of multi-token chunks and Unit-length chunks. In 2009, the BILOU scheme was proved that it outperformed the performance of the BIO scheme (Ratinov, & Roth, 2009).

In 2003, Florian et al., proposed a framework which used combination classifier. They used four classifiers: Robust Risk Minimization Classifier, Maximum Entropy Classifier, Transformation-Based Learning Classifier, and HMM Classifier. Researchers set different weights for each classifier and finally found that the combined classifier based on the RRM algorithm is the optimal combination (Florian, Ittycheriah, Jing, & Zhang, 2003).

In several multi-class SVM building methods were proposed by Hsu and Lin (Hsu & Lin, 2002). Then, Yamada et al. (Yamada & Matsumoto, 2002), used the one-vs-rest method built the first SVM-based NER system (Isozaki & Kazawa, 2002). Isozaki and Kazawa use the quadratic kernel to deal with the NER task. They also made an XQK feature selection method to remove the useless features to improve the performance of the SVM based model (Isozaki & Kazawa, 2002). Kazama et al., chose the pairwise method proposed by Krebel (Krebel, 1999) to build the SVM model and they introduced a new class splitting technique based on part-of-speech. Shen et al., proposed the use of the informativeness measure and similarity measure for named entity and proposed two combined strategies (Shen, Zhang, Su, Zhou, & Tan, 2004). In terms of the characteristics of the Biomedical NER, Lee et al., developed a two-phase model based on SVMs, it first identified a word as part of an entity or not and then identify the particular type (Lee, Hwang, Kim & Rim, 2004).

2.3.4.3 Neural network based methods

Deep learning is used in computing and modelling multiple levels of abstraction data through multiple processing layers, and then output people expected results. It uses

backpropagation algorithms to adjust the parameters in the neural network to continuously approximate the optimal model. Deep learning is widely used in computer vision, artificial intelligence, genomics, and biomedical science, and other domains and brought a breakthrough in image, video, audio and speech processing (LeCun, Bengio & Hinton, 2015).

Using the unsupervised word representation as an extra word feature is a simple approach to improve the performance of supervised NER (Turian, Ratinov, & Bengio, 2010). Traditionally the supervised methods represent a word as one hot vector. The length of the vector is the size of the vocabulary and is only based on one dimension. The one-hot representation cannot deal with an unlabelled word. Also, every word is one dimension that is independent with each other, and it cannot represent the relationship between the words. In order to improve performance, new word representation techniques are used for many NLP tasks (Tang, Cao, Wang, Chen, & Xu, 2014).

Word representation is divided into three categories, distributional representation, clustering-based word representation, and distributed representation (Turian, Ratinov, & Bengio, 2010). In 2014, Tang et al., used three word representation methods and applied them into a CRF based biomedical NER. They found that word representation can individually improve the biomedical NER system, they also found that different word representation methods can work together and further enhance the performance of the system (Tang, Cao, Wang, Chen, & Xu, 2014).

Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) are based on the theory of distribution hypothesis, using the statistical method to reduce the high dimensionality co-occurrence matrix to the low dimensionality latent semantic matrix to obtain the semantic representation of words. This is since the distribution hypothesis considers that words with similar contexts have similar semantic information. Under this representation, the semantic similarity of two words can be directly translated into the spatial distance of two word representation vectors. In 2009, Guo et al., constructed a Weakly Supervised Latent Dirichlet Allocation (WS-LDA) method used for NER in query (NERQ). For the characteristic of NERQ, the WS-LDA method uses a probabilistic approach to handle the ambiguity of named entities. It can make sure there is alignment between the latent tags and the predefined tags (Guo, Xu, Cheng, & Li, 2009). In 2015, Konkol et al., researched the effect of using Hyperspace Analogue to Language (HAL), Random Indexing (RI), and Latent Dirichlet Allocation (LDA) as features in NER and they also compared the semantics features with commonly used features (Konkol, Brychcín, & Konopík, 2015).

Clustering-based word representation based on the similar semantic word has the same or a close cluster. The representation approach induces unlabelled word as clusters, and represents words as a cluster that a word belongs to. In 2011 Chrupala compared Brown Clustering with the Clustering word representation feature and the LDA feature of NER

task. The result showed the Brown Clustering needed a large number of classes to avoid ambiguous classes and a longer time was required to train the Clustering model (Chrupala, 2011).

In 2003, Bengio et al., proposed the first distributed representation method. Through the feedforward Neural Network Language Model (NNLM) it trained the data, and represented a word as a vector (Bengio, Ducharme, Vincent, & Jauvin, 2003). The model contains four layers, the input layer, projection layer, hidden layer, and the output layer. It learned word vector representation and statistical language modelled through a linear projection layer and a non-linear hidden layer (Mikolov, Chen, Corrado, & Dean, 2013). Distributed representation, also called word embedding (Turian, Ratinov, & Bengio,2010), is similar to distributed representation methods, and it also based on the distribution hypothesis. The core is a context based representation and the relationship between the context and the target words. Compared with one-hot representation, each element of the word embedding vector is changed from an integer to a float, and the range is in a real number; the original sparsely high dimension space is compressed into the space of a smaller dimension. Word embeddings are typically using for neural language models (Turian, Ratinov, & Bengio, 2010).

In 2008, Collobert and Weston proposed the C&W model. The model is used to complete multiple NLP tasks by one model (Collobert & Weston, 2008). The model not only applies to word embedding it also uses the word embedding feature to feed it into the convolutional network final that used the Viterbi algorithm to predict the word tags (Collobert, 2011). In 2011, Collobert improved on the C&W model. He used Parse Tree to represent the word and to separate the lower case word and the capitalization word as two features (Collobert, 2011).

There is another language model that is called the hierarchical log-bilinear (HLBL) model. It combines the log-bilinear model (Mnih & Hinton, 2007) and the hierarchical evaluation technique (Mnih & Hinton, 2009). The method uses a hierarchical structure for the final prediction. The last network layer is a tree structure, each non-leaf node is used to classify the prediction vector. Finally, the leaf node can determine the tag of a word. The M&H method uses the bootstrapping method to build the tree. Starting with a random tree, it can continuously adjust and iterate based on the classification results. The final result is a balanced binary tree. The HLBL can also be used for the word embedding task.

In 2010, Turian et al., compared the effects of the C&W word vector and the HLBL word vector as auxiliary features. The result showed that at NER, the impact of C&W vectors have a slight advantage (Turian, Ratinov, & Bengio, 2010).

In 2013, Mikolov et al., proposed the Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model. The tool is called Word2vec. The CBOW model uses one hot representation of the context of the target word as input. Then the input vector is respectively multiplied by the coefficient matrix to obtain the hidden layers. It is

necessary to calculate the average of hidden layers through the linear activate function and then multiply it by another coefficient matrix to obtain the output layer. The vector of the output layer needs to be compared with the one hot representation of the target word to calculate the loss. Some of the impossible words are filtered out by the Hierarchical Softmax method based on Huffman coding and then negative sampling is used to remove some negative sample words. Thus, the time complexity of the CBOW model will decrease. The Skip gram training process is similar, except the input and output are opposite (Mikolov, Chen, Corrado, & Dean, 2013) (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

After this, the neural network method is widely proposed to deal with the NER. The traditional NER method needs hand-designed features and then is fed to a classification algorithm. Instead, Collobert et al., used word embedding layers to extract the features from large quantities of unlabelled text and then they fed the features to the deep network trained by backpropagation (Collobert et al., 2011).

However, there are two main limitations, the first is that their neural network still faced a long-term dependencies problem, the simple feed- Forward neural network only limits the context to a fixed sized window, and the long-distance relations between the words are ignored. Second, the word embedding cannot extract the character level features (Chiu & Nichols, 2016).

In order to gain word morphology and shape and capture intra-word features, in 2014, Santos and Zadrozny added a convolutional layer to extract the character-level representations and they combined it with word-level representations (Santos & Zadrozny, 2014). This neural network produces a higher quality of POS tag. In 2015, Santos and Guimaraes popularized this neural network architecture to the NER, and it extended the neural network proposed by Collobert et al., in their study (Santos & Guimaraes, 2015).

The Long short-term memory (LSTM) model can be effective in terms of managing the processing with long-term dependencies problem. In 2015, Huang et al., applied the bidirectional LSTM-CRF model to manage the sequence tagging problem. They also, compared the LSTM model, the bidirectional LSTM model, the LSTM-CRF model, and the bidirectional LSTM-CRF model. The result showed that the bidirectional LSTM-CRF model got the highest performance of the four models. In the bidirectional LSTM-CRF model, the CRF layer can efficiently predict the current tag through the past and future information provided by the forward and backward algorithm of Bi-LSTM model (Huang, Xu, & Yu, 2015). In 2016, Lample et al., compared the bidirectional LSTM with a conditional random layer (Bi-LSTM-CRF) and LSTMs with states represented by a stack (s-LSTM). Based on the above comparison, they also compared the char-level word embedding and non-char-level word embedding models. The results showed that the char-level word embedding Bi-LSTM-CRF model gained the best performance (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016).

In 2016, Chiu and Nichols used a convolutional neural network as a character level features extractor and combined it with the Bi-LSTM network model (Chiu & Nichols, 2016). The model used a convolution and max layer to extract the feature from pre-training vectors. In their best model, they used the additional features capitalization feature, lexicons, and a four-dimensional vector representing four character types.

Previous systems generally use a neural network to enhance the distributed representations rather than completely replace it. If the model entirely depends on neural embeddings without any handcraft features, the performance of the systems decreased rapidly. In order to solve this problem, Ma and Hovy designed a real end-to-end system (Ma & Hovy, 2016). The system operates without any feature engineering and data pre-processing. They use CNN computed character representation combined with word embedding and then feed it into the Bi-LSTM network, and finally they use the CRF layer to predict the tags.

Recently, pre-trained word vectors have become an essential part of the neural network structure (Peters, Ammar, Bhagavatula, & Power, 2017). The pre-trained word embedding learning from the unlabelled data, is then generally added into the neural network as additional features. If it is combined with the word embedding of the training data as input, it can increase the performance of the sequence tagging model. Peters et al., proposed using the bidirectional language models to train the pre-train word vectors and proved the pre-train word vectors can improve the mainstream neural network model (Peters, Ammar, Bhagavatula, & Power, 2017).

In 2018, Peters et al., proposed a new language model called ELMo (Peters et al., 2018). It uses two independently trained multilayer Bidirectional LSTM structures, the word level and the character level embedding vectors as input, and. the output embedding vector related to the contexts. The model is not only used for NER it is also used for many downstream tasks (Devlin, Chang, Lee & Toutanova, 2018).

In 2016, Rei et al., used the attention mechanism and added it into the neural sequence labelling models (Rei, Crichton, & Pyysalo, 2016). The model can dynamically choose how to combine the features from word embedding and from the character-level component through the attention mechanism. The attention mechanism based feature selection architecture can align two word representations and gain extra benefit from both the word-level and the character-level embedding. Bharadwaj et al., added the attention model to the Bi-LSTM-CRF model. They used the International Phonetic Alphabet (IPA) and word embedding as features. Their paper proved that the attention model and the IPA improves the statistical efficiency of the model (Bharadwaj, Mortensen, Dyer, & Carbonell, 2016).

Pre-training vectors that generally come from the text are unrelated to the current task. Through the pre-training vectors, models learn generally effective representations rather than representations of the current task and this is a key problem of the Pre-training model

(Clark, Luong, Manning, & Le, 2018). Thus, Clark et al., proposed a Semi-Supervised learning method called Cross-View Training. The method uses different approaches to train the labelled data and the unlabelled data. The method trains the labelled data as standard for the supervised learning processing; for the unlabelled data, the method uses a primary prediction module plus multiple auxiliary prediction modules. The output of the auxiliary prediction modules is fitted to the output of the primary prediction module while training on the unlabelled data, and the encoder portion is shared. The input to each prediction module is different, the primary prediction module is the complete input, and the input to the auxiliary prediction module is a subset of the complete input. CVT can improve the "representation" of the model. The Auxiliary Prediction Module can learn from the predictions of the primary prediction module because the primary prediction module has better input with an unrestricted perspective. Although the input of the auxiliary module is a restricted input sample, they are still learning something from the primary prediction module, thus the quality of the "representation" is improving. This improves the entire model because they share the encoder (Clark, Luong, Manning, & Le, 2018).

In 2018, Google AI Language proposed the BERT model. It is a new language representation model. The BERT model can be used for eleven downstream tasks. In the model, the input is combined with token embeddings, the segmentation embeddings, and the position embeddings. The input part is a linear sequence. Token embeddings represent the word vector, the start word is "CLS", which can be used for the classification tasks; the special character [SEP] is a symbol that is used to split two sentences. The Segment embeddings are used to distinguish between the sentences. Position embeddings indicate location information. The NER task belongs to the token level, and it classifies the corresponding position of every word in the last transformer layer (Clark, Luong, Manning, & Le, 2018). The BERT model achieved the state-of-the-art result of the NER.

2.4 Hidden Markov Model (HMM)

The HMM is a statistical model that is used to describe a Markov process with hidden unknown parameters. It determines the hidden parameters of the process through the sequence of observations and it then uses these parameters for further analysis (Rabiner, 1989).

Figure 1 below shows the structure of HMM, X is a hidden variable that observer cannot know. $x(t)$ represents a state at time t. Each observation variable y only depends on $x(t)$, $x(t)$ is related to the previous state $x(t-1)$.

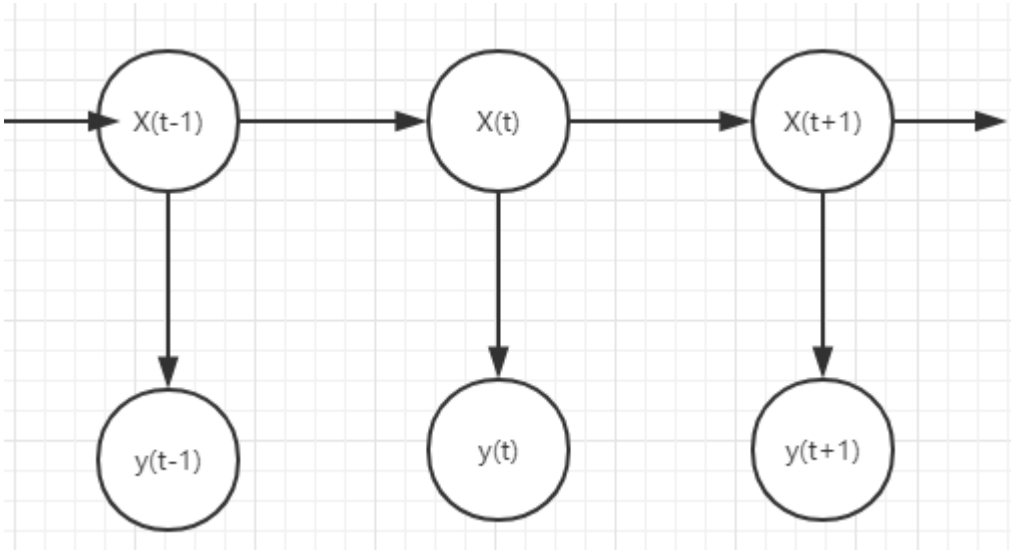

Figure 1: The structure of HMM

If the hidden states have N values, then at time t the hidden state can be one of the N values. Thus, at time $t+1$, there are also N possible values for the hidden states. This means that there is a total of N^2 probabilities from a hidden state transit to the next hidden state. For observation variable y it has M possible values, each value of hidden state have probabilities that emit to every observation variable. Therefore, if the observed sequence is represented as Y, the hidden state sequence is represented as X, and the X and Y could be represented as the following equations:

$$Y = \big(y(0), y(1), \cdots, y(n)\big)$$

$$X = \big(x(0), x(1), \cdots, x(n)\big)$$

The probability of the observed sequence Y through HMM could be represented as the following equation:

$$\mathrm{P(Y)} = \sum_{X} P(Y|X)P(X)$$

For the unlabelled data, because of the existence of the hidden variables, the analytical solution of the parameters cannot be obtained directly. The Expectation Maximization (EM) algorithm is used to iterate until convergence to obtain the model parameters. The EM algorithm is divided into two parts: the E step and the M step. In the E step, it uses the known parameters to obtain the posterior distribution of the hidden variable $P(T|S, \theta^{old})$; in step M, it calculates the expected maximum of the log-likelihood under this posterior distribution. The expectation is a function of parameter θ, and it maximizes the expectation function $Q = (\theta, \theta^{old})$, so it can get the solution of θ, and then it uses this solution as the new θ^{old} to bring it into the E step, and then it iterates until the final convergence (Moon, 1996) (Zhang, Brady & Smith, 2001).

After training, using the HMM model it can predict new sentence sequence. If it is given a set of observation sequences, it will find the most likely hidden sequence that corresponds by using the Viterbi algorithm. (Ghahramani, 2001)
The HMM model is a very common statistical method, and it has been used to solve many NLP problems, such as, speech recognition, machine translation, POS tagging, NER, and so forth (Ponomareva, Pla, Molina, & Rosso, 2007).

2.5 Support Vector Machine (SVM)

The SVM is a popular machine learning method for dealing with classification, regression, and distribution estimation problems (Chang & Lin, 2011). It was proposed by Cortes and Vapnik in 1995. The purpose of the SVM model is to find a hyperplane to segment the sample. The principle of segmentation is to maximize the interval. The model finally transforms it into a quadratic programming problem (Suykens & & Vandewalle, 1999).
The SVM model is to find hard margin maximization to determine the optimal splitting plane to split the two kinds of data if the data is linearly separable. The Hard margin means the distance from the support vector to the split plane. The support vector is the closest hyperplane to the split plane, which is also half the distance between the two categories of data (Burges, 1998). The diagram is shown below.

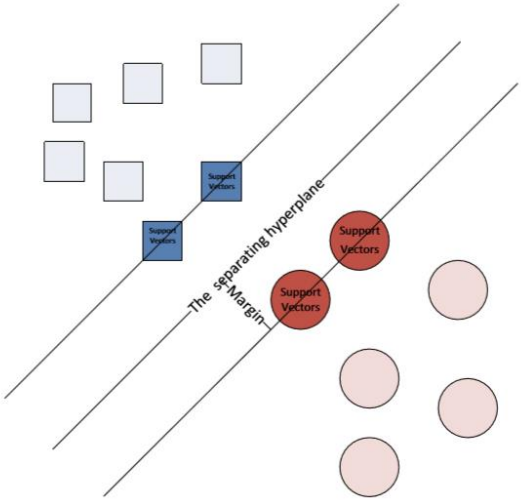

Figure 2: The structure of SVM

Although there are countless lines (planes) that can separate the samples, the SVM finds the line (plane) that keeps the interval maximized.
If the data is not linearly separable, the splitting plane cannot completely divide all of the data, whereas the linear splitting method can ensure that most of the data is correctly classified. Thus, SVM needs to achieve Soft Margin maximization. The Soft Margin is if the SVM is not linearly separable, and the support vector is not the closest vector to the split plane. The SVM cannot keep the margin maximization at this time, it needs to make

the transition band wide enough.

In actual data, usually the data is linearly inseparable, and using a straight line cannot separate the two types of samples, however by using a nonlinear model it can separate them. Therefore, nonlinear transformation can be used to transform nonlinear problems into linear problems (Smola & Schölkopf, 2004).

For this problem, the training samples can be mapped from the original space to a higher-dimensional space by kernel function, so the samples are linearly separable in higher-dimensional space (Ju, Wang, & Zhu, 2011). If the number of attributes is finite, then the original spatial dimension is finite so there must be one hyperplane that makes the sample separate in a high dimensional feature space. The training sample can be translated to $(x_i, y_i),\ i = 1, \cdots, n,\ x \in R^d,\ y \in \{+1, -1\}$ (Isozaki & Kazawa, 2002). The hyperplane can be expressed as: $f(x) = w^T \varphi(x) + b,\ \varphi(x)$ is feature vector after mapping x.

2.6 Conditional Random Field (CRF)

The CRF is a sequence modelling framework similar to HMM. It has all of the advantages of HMM and avoids the label bias problem of MEMM (McCallum, Freitag & Pereira, 2000). The goal of the model is to learn the mapping function $x_s \rightarrow y_s$ allows the correct output label maximization. However, each output y_s is not independent. The CRF model is able to predict an output vector $\mathrm{y} = y_0, y_1, \cdots, y_t$ through calculating the conditional probabilities of the random variables given an observed feature vector $x = x_0, x_1, \cdots, x_t$ (Sutton & McCallum, 2012). The CRF combines discriminative classification and graphical modelling. So, it has the abilities of compactly modelling and it uses a huge number of input features for the prediction (Sutton & McCallum, 2012).

The CRF model is widely applied to many fields. In computer science, the CRF model has been applied to text and speech processing, such as, part-of-speech (POS) tagging (Ghosh, Ghosh & Das, 2016), NER (Seker & Eryiğit, 2017), information extraction (Ebersbach, Herms, Lohr & Eibl, 2016), and syntactic disambiguation (Junaida, Jayan & Sherly, 2017). In bioinformatics, the application includes protein alignment (Morales-Cordovilla, Sanchez & Ratajczak, 2018), and RNA secondary structure prediction (Johansen, Sønderby, Sønderby & Winther, 2017). In computer vision, the application includes object extraction (Li, Femiani, Xu, Zhang & Wonka, 2015), and image segmentation (Liu, Lin & Shen, 2015).

For the CRF model, $G = (V, E)$ is a graph. V represents the node set, E represents the line set which is a connection between the two nodes. Each node v assigns random variables Y_{v} so $Y = (Y_{\mathrm{v}})\ v \in V$, every Y_{v} follows the Markov property so the grapy can be translated to the equation (Lafferty, McCallum & Pereira, 2001):

$$P(Y_{\mathrm{v}} | X, Y_{\mathrm{w}},\ w \neq v) = P(Y_{\mathrm{v}} | X, Y_{\mathrm{w}},\ w \sim v)$$

$w \sim v$ means w and v are connected in G.

However, in this graph G, there is no direction between the two nodes. Thus, in the NLP domain, researchers used linear chains CRF (Varma, Krishnamoorthy & Pisipati, 2016) (Taskar, Abbeel & Koller, 2002).
For NER, the linear chains is a popular method for CRF. The structure is shown below:

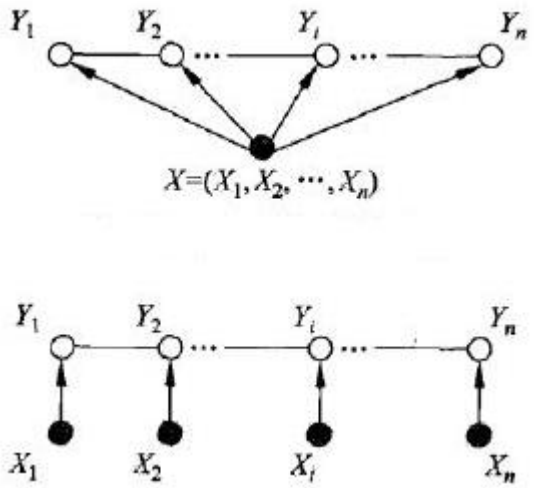

Figure 3: The structure of linear chains CRF model

Nodes are composed of a linear chains structure, the nodes chains correspond with sequence Y. If a random variable sequence $X = X_1, X_2, \cdots, X_n$ and $Y = Y_1, Y_2, \cdots, Y_n$ with a linear chain structure and the conditional probability distribution of the random variable sequence Y is $P(Y|X)$ it obeys the Markov property. The equation is:

$$P(Y_i|X, Y_1, Y_2, \cdots, Y_n) = P(Y_i|X, Y_{i-1}, Y_{i+1})$$

Then by giving it the observation sequence X, we can use the conditional probability distribution $P(Y|X)$ of the random variable sequence Y to predict the Y sequence (Wallach, 2004).

2.7 Deep Learning

Deep learning is used to learn multiple levels of abstraction data through a model that has multiple processing layers, and output results that people expect. It uses backpropagation algorithms to adjust the parameters in the neural network to continuously approximate the required model. Deep learning is used in computer vision, artificial intelligence, genomics, and biomedical science, and so forth, and in other domains and has brought a breakthrough in many tasks (LeCun, Bengio & Hinton, 2015). Deep learning has gained significant success in supervised learning, unsupervised learning, and semi-supervised learning (LeCun, Bengio & Hinton, 2015).
In 2006, the problem of gradient vanishing in deep learning has been investigated. Hinton et al., proposed a solution for the gradient disappearance problem in deep network training: by using unsupervised pre-training it initializes the weights and then fine-tunes the

parameters through supervised training. The main idea is to learn the structure of the training data through the self-learning method, and then refines it through supervised training (Hinton & Salakhutdinov, 2006).

The deep neural network is not necessarily applicable for training because the input of each layer will change during the training. It makes it challenging to gain the saturating nonlinearities model (Schmidhuber, 2015). In general, the input data is randomly divided into several data chunks from the entire input data. Feeding the chunked data into the neural network can make the model more generalized. In 2015, Ioffe and Szegedy proposed that normalizing each batch of data can improve the neural network. During the training model, the weights are changed after every gradient descent. Thus normalizing the batch data can ensure the next processing layer will gain similar data compared with the previous distribution (Ioffe & Szegedy, 2015).

Stochastic Gradient Descent (SGD) and its variants are probably the most widely used optimization algorithm in general deep learning (Goodfellow, Bengio & Courville, 2016). It is necessary to find the minimum loss in the neural network, because the predicted value is closer to the actual tag value if the loss value is smaller. Actually, the SGD algorithm involves Mini-batch gradient descent. It repeats the training from the same batch size of random samples from the training set until it finds the optimal point. The Mini-batch gradient descent can significantly reduce the number of iterations that are required of convergence, and this saves training time (LeCun, Bengio & Hinton, 2015). However, gradient descent has two problems. The first one is the parameter tuning that will be very slow if the value is close to the minimum or plateau area. The second problem is the value cannot converge to the global minimum, it can only converge to the local minimum. On the other hand, the Multi-layer neural networks are not convex functions, so theoretically there are multiple local minima, and as the number of layers increase, the local minimum will increase. In order to accelerate the SGD algorithm, momentum is added into SGD (Ruder, 2016). Once the gradient falls to a local minimum, it is possible to use the momentum to cross the local minimum. In 2013, Sutskever et al. (Sutskever, Martens, Dahl & Hinton, 2013), proposed the Nesterov accelerated gradient (NAG). Before the gradient direction change, the NAG can get information in advance, thus reducing useless iterations (Ioffe & Szegedy, 2015). Dean et al., used the Adagrad algorithm for training large-scale neural networks (Dean et al., 2012). Different parameters require a different learning rate. The Adagrad algorithm can give different weights to the learning rates. Thus, the learning rate can adapt to parameters that have been found (Duchi, Hazan & Singer, 2011). In 2015, Kingma et al., proposed the self-adapting learning rate for each parameter (Kingma & Ba, 2014). The self-adapting learning rate can make the gradient jump out of a local minimum. Dozat proposed a variant of Adam in 2016, this variant method added NAG into Adam (Dozat, 2016).

In 2012, Krizhevsky et al., proposed AlexNet, which is a deep convolutional neural

network (Krizhevsky, Sutskever, & Hinton, 2012). It is a breakthrough in deep learning, and the model detonated a wave of neural network applications and won the 2012 image recognition contest, making CNN the core algorithm model for image classification (Minar & Naher, 2018). AlexNet contains convolutional layers and three fully connected layers. It uses Graphics Processing Units (GPU) to accelerate the operation. It also uses the Rectified Linear Unit (Relu) as the activation function. The Relu method allows outputs of some neurons that are zero, which cause the sparseness of the network, and it dramatically increases the speed of calculation. In deep neural network models, if a model increases the N layers, theoretically the activation rate of ReLU neurons will decrease 2^n. Also, according to Jarrett et al., the Relu method can reduce the interdependence of parameters, and this alleviates the occurrence of the over-fitting problem (Jarrett, Kavukcuoglu & LeCun, 2009).

The initialization of the network is very important for deep neural networks. In 2010, Glorot and Bengio proposed the normalized initialization method (Glorot & Bengio, 2010). He et al., proposed an initialization method, which is more suitable for neural networks using ReLU. Also, compared with the initialization method that Glorot and Bengio proposed, the method He et al., proposed can rectify extremely deep neural networks (He, Zhang, Ren & Sun, 2015).

In 2012, Hinton et al. (Hinton, Srivastava, Krizhevsky, Sutskever & Salakhutdinov, 2012), proposed the dropout method. If a complex feedforward neural network is trained in a small data set, it is easy to cause overfitting. If it randomly drops neurons and their connection during training processing it can prevent overfitting. During testing, in order to compensate for the different numbers of neurons between the testing and the training, the network needs to rescale the weights (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014). In 2012, Krizhevsky et al., used the Dropout algorithm in AlexNet to prevent overfitting (Krizhevsky, Sutskever, & Hinton, 2012).

After this, many researchers proposed different CNN architectures. In 2016, Redmon et al. proposed YOLO (You Only Look Once) which is a CNN architecture use for unified and real-time object detection (Redmon, Divvala, Girshick & Farhadi, 2016). Max-Pooling Convolutional Neural Networks (MPCNN) (Krizhevsky, Sutskever & Hinton, 2012) was proposed in 2012, it is used for processing image or object detection. Region-based Convolutional Neural Network (R-CNN) (Girshick, Donahue, Darrell & Malik, 2014), Fast Region-based Convolutional Network (Fast R-CNN) (Girshick, 2015), Faster Region-based Convolutional Neural Networks (Faster RCNN) (Ren, He, Girshick & Sun, 2015), Mask Region-based Convolutional Network (Mask R-CNN) (He, Gkioxari, Dollár & Girshick, 2017), Multi-Expert Region-based Convolutional Neural Networks (ME R-CNN) (Lee, Eum & Kwon, 2017) are proposed successively. Deep learning is widely used in the field of image processing and object detection.

In 2003, a neural network language model based on the feed-forward neural network had

been proposed, which could be used for learning a distributed representation for words. Nowadays, this distributed representation for words is called word embedding. These word embedding vectors as a feature can feed into hidden layers (Bengio, Ducharme, Vincent & Jauvin, 2003). The new deep learning technique pushes up the character-level and word-level based model moving forward even modelling a single syllable of Unicode characters (Sutskever, Martens & Hinton, 2011). Recurrent neural networks (RNNs) (Mikolov, Karafiát, Burget, Černocký & Khudanpur, 2010) and long short-term memory networks (LSTMs) can solve long term dependency problem. In NLP, feed-forward neural networks have been replaced with RNNs and LSTMs. However, the RNNs is difficult to train due to the gradient vanishing and exploding problem (LeCun, Bengio & Hinton, 2015).

Through sharing parameters between models and weights of different layers, a neural network can train on multiple tasks in one model. Multi-task learning has been used in many fields, such as, computer vision (Girshick, 2015), NLP (Collobert & Weston, 2008), biomedical drug discovery (Ramsundar et al., 2015) and so forth. In 2008 Collobert and Weston applied the multi-task neural network for the NLP domain (Collobert & Weston, 2008). Their model shared word embedding matrices between the two models and trained different tasks.

In 2013, word2vec was proposed (Mikolov, Chen, Corrado, & Dean, 2013) (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). It was developed specifically to manage word embedding. It used Continuous bag-of-words and skip-gram architectures training on a very large corpus.

In 2014, Sutskever et al., proposed the sequence-to-sequence framework (Sutskever, Vinyals & Le, 2014), it maps one sequence to another one using neural networks. In the framework, the neural network encoder processes the input data and compresses it into a vector representation; then the neural network decoder predicts the output symbol by symbol, based on the encoded state. This framework is very powerful, not only for natural language generation task it also can deal with any conditions on a sequence. Vinyals et al. through the sequence-to-sequence framework built an image-caption generator (Vinyals, Toshev, Bengio & Erhan, 2015).

Attention (Bahdanau, Cho & Bengio, 2014) proposed by Bahdanau et al., is widely applicable and potentially useful for any task that requires making decisions based on input. The input does not have to be a sequence, the input can consist of any other representations, such as, an image (Xu et al., 2015).

2.8 Deep Neural Network

The Deep neural network (DNN) consists of an input layer, several hidden layers and an output layer, which enables it to extract features from the input layer, process various

functions in hidden layers and output the expected results (Minar & Naher, 2018). Each layer is composed of many layers that are connected by simple processor units. These units are called neurons. Through assigning the weight between each neuron and each layer it makes the network display the desired behaviours (Schmidhuber, 2015). DNN shows not only good performance in supervised learning it is also very successful in supervised learning, semi-supervised learning, and reinforcement learning (LeCun, Bengio & Hinton, 2015).

Typically, supervised learning is applied for a large labelled data set (Nadeau, 2007). Supervised learning based on DNN is widely used in many domains, such as, speech recognition (Hinton et al., 2012), image classification (Krizhevsky, Sutskever & Hinton, 2012), medical research (Esteva et al., 2017) and so forth.

Unsupervised learning learns features from unlabelled data through feature selection layers. Typically, unsupervised learning is applied for data without handcrafted features and data with limited labels. This learning method is useful for object detection (LeCun, Bengio & Hinton, 2015). The features extracted through the unsupervised learning network are widely used for image processing and for the NLP domain.

Semi-supervised learning combines supervised learning and unsupervised learning, and it uses the combination of the label data and the unlabelled data to train models. It is very useful if the labelled data is very expensive or if it is not available (Zhu, 2005). It can also, generalize from the small labelled data to that large unlabelled data (Kingma, Mohamed, Rezende & Welling, 2014). In 2014, Kingma et al., proposed a framework semi-supervised learning method through the use of deep neural networks and probabilistic modelling (Kingma, Mohamed, Rezende & Welling, 2014). In 2016, the deep neural network based semi-supervised learning was used in the image processing domain (Kipf & Welling, 2016).

2.9 Convolution Neural Network (CNN)

Back Propagation (BP) was proposed by Rumelhart and Hinton et al., (Rumelhart, Hinton& Williams, 1988). LeCun et al., used the BP algorithm to train the multi-layer neural network to identify handwritten zip codes (LeCun et al., 1989). After that, LeCun proposed the LeNet5 in 1998 (LeCun, Bottou, Bengio & Haffner, 1998). It contains the basic modules of deep learning, the convolutional layer, the pooled layer, and the fully connected layer. The model includes the basic concepts of the CNN model. It uses convolution layers to extract spatial features, it uses the Tanh or Sigmoid algorithm as a nonlinear optimizer, it chooses the Multilayer Perceptron (MLP) as the final classifier and it avoids large computational costs by the sparse connection matrix between the layers. In 2012, Alex et al. (Krizhevsky, Sutskever, & Hinton, 2012), proposed AlexNet. It shows that CNN can learn more complex object levels if it works on a larger dataset. The model

uses ReLU as an activation function and it uses Local Response Normalization (LRN) to increase generalization. The AlexNet uses the largest pooling to avoid the average pooling blurring. The data augmentation and dropout method are also used in AlexNet. It randomly extracts the 224×224 size image from the 256×256 original images, which is equivalent to an increase of 2048 times of the original dataset. The AlexNet uses the dropout method in the next three fully connected layers, randomly ignoring part of the neurons to avoid over-fitting. It can also accelerate the network training through GPU. In 2013, Sermanet et al. (Sermanet et al., 2013), improved AlexNet and proposed the learning bounding box. They used a 3×3 filter and a 2×2 maximum pooling filter continuously deepening the network structure to improve performance. Using the stacked small filter is better than using the large filter because the multiple layers of a nonlinear layer can increase the depth of the network to ensure that more complex modes are learned. The model also requires fewer parameters. In 2015, He et al., proposed the ResNet (He, Zhang, Ren& Sun, 2016). As the network deepens, the accuracy of the training set decreases and the error rate increases. Because the model is too complicated, the optimization becomes more difficult. The model cannot achieve good learning results. The ResNet use a shortcut connection to skip one or more layers. If the layers of the model are deepened, this structure can solve the degradation problem. It increases the training speed of the model, and this improves the training effect.

2.10 Recurrent Neural Network (RNN)

Since the Convolutional neural network developed, many researchers found it difficult to model the changes of time sequence. In the traditional neural network model, the layers are fully connected, the neurons between each layer are disconnected, the signals of the neurons can only be propagated to the upper layer, and the processing of samples is independent at each moment. Therefore, this neural network is powerless for many problems. For example, predicting named entity category of a word generally requires preceding words, because the word is not independent of the previous words and the later words in a sentence. The current output of a Recurrent Neural Network (RNN) (Goller & Kuchler, 1996) is related to the previous output. The network stores the previous information in the internal network and applies it to the calculation of the current output. Thus, the nodes between the hidden layers are connected, and the current input of the hidden layer contains not only the output of the input layer at the current moment it also contains the output of the hidden layers at the previous moment. In theory, a recurrent neural network can process the sequence data of any length, yet, in order to reduce complexity, it often assumed that the current state is only related to the one previous state. Nowadays, RNN is widely used in the NLP domain, such as, language modelling (Mikolov, Karafiát, Burget, Černocký & Khudanpur, 2010), generating Text (Sutskever,

Martens & Hinton, 2011), and in machine translation (Liu, Yang, Li & Zhou, 2014). It is also used to generate image descriptions (Karpathy & Fei-Fei, 2015).

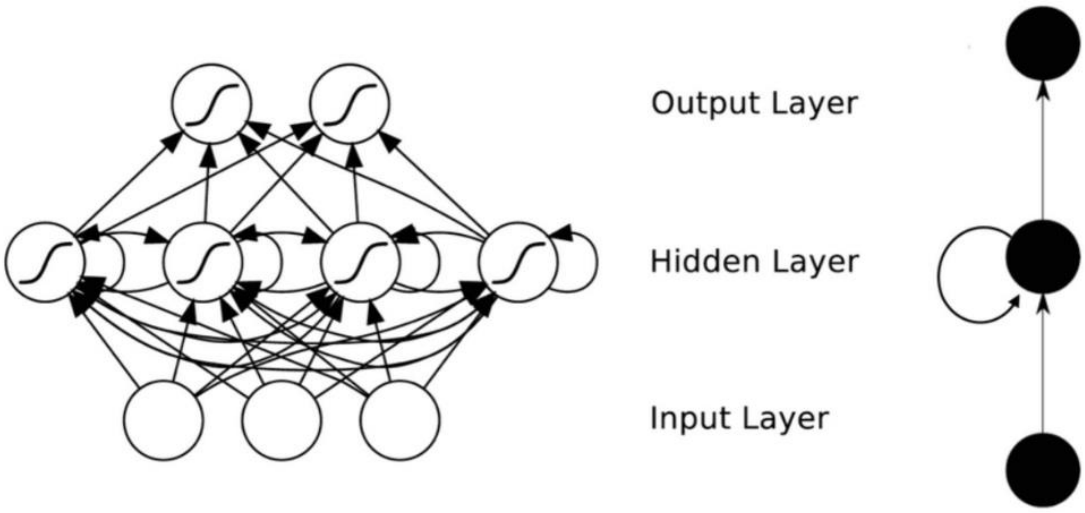

Figure 4: Overall structure of RNN

Typical RNN consists of input layers, hidden layers, and output layers, as Figure 4 shows above. For time t, the hidden layer follows the structure shown below.

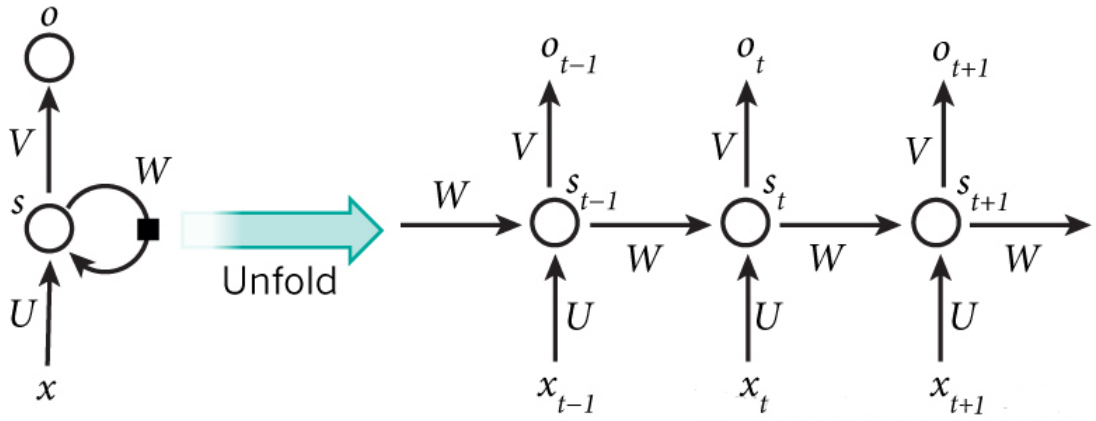

Figure 5: Structure of RNN at time t

For the left of Figure 5, x, s, and o represent the input layer, the hidden layer, and the output layer. U is the weight matrix for the input of the hidden status. V is the weight matrix for the hidden status to output. W is the weight matrix for the hidden status to the next hidden status.

See the equations below for the output of the Hidden layer and the output of the output layer at time t. The activation function is function f, for example, sigmoid, ReLU, or Tanh, and so on. Function g is Softmax function (Mikolov, Karafiát, Burget, Černocký & Khudanpur, 2010).

$$h_t = x_t + Ws_{t-1}$$
$$s_t = f(h_t)$$
$$o_t = g(Vs_t)$$

h_t is an input value that transforms through weight matrix U and W.
In order to use future contexts in the sequence label tasks, the Bi-directional Recurrent Neural Network combines the forward and the backward hidden layer.

2.10.1 Long Short Term Memory (LSTM)

Although RNN works well for time sequence tasks, it faces gradient vanishing and exploding problems (Bengio, Simard & Frasconi, 1994). In 1997, the long short-term memory (LSTM) was proposed by Hochreiter and Schmidhuber Hochreiter and Schmidhuber, 1997). It is very suitable for sequence modelling (Tang, Qin & Liu, 2015). LSTM is a variant of RNN (Pascanu, Mikolov & Bengio, 2013) which uses the enhance module instead of the recurrence unit of RNN (Pham & Le-Hong, 2017). Thus, LSTM can access more contextual information than RNN. Gers & Schmidhuber (Gers & Schmidhuber, 2000) proposed the Peephole structure to increase the performance of LSTM. It lets the gate layers accept the input of the cell state. Another variant with a more significant change is the Gated Recurrent Unit (GRU), which Cho et al., proposed (Cho, Van Merriënboer, Bahdanau & Bengio, 2014). It places the forget gate and the input gate so they are combined into a single update gate.

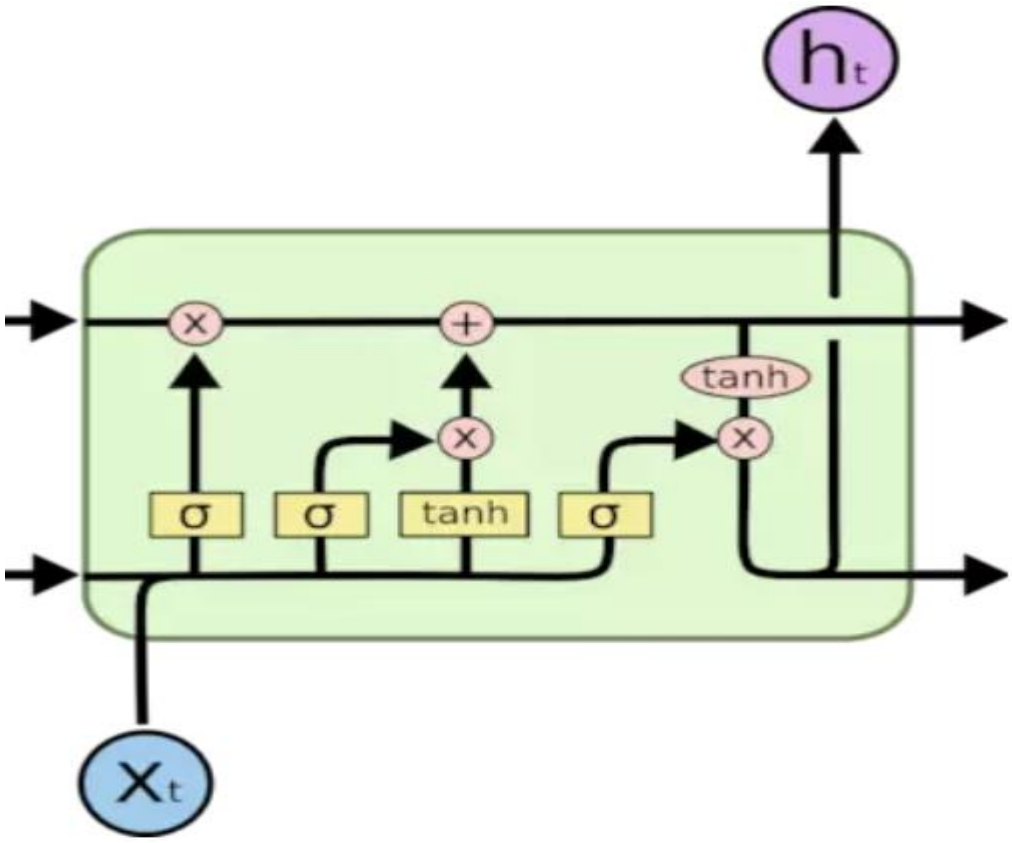

Figure 6: Structure of the memory cell unit

The memory cell unit module consists of the input gate, the forget gate, and the output gate. The cell state and hidden state are mixed. The structure of the memory cell unit is shown above.

The first step of the LSTM is deciding whether to discard information or not through the forget gate. The input of the forget gate includes the previous hidden states and the present input information. Then the output is a value between 0 to 1. The value transmits to the state of the memory cell. 0 represents a complete discard, and 1 represents a complete hold. Then the input gate determines the information update, and a new candidate vector will hold it into the memory cell. The next step is the update state of the cell, it throws the information that the forget gate determined, and it adds the candidate vector. In the end, the output is determined by the cell state and the output gate.

In order to use the future contextual information, it is necessary to consider the contexts from two directions. The Bidirectional LSTM (Bi-LSTM) consists of two LSTMs which uses two hidden states layers to gain the past and future information through a forward and backward sequence. The output is determined by the state of the hidden layers of the two LSTMs (Graves, Mohamed & Hinton, 2013).

2.11 Summary

This Chapter mainly describes the NLP and NER. In terms of NER, we researched it from the aspect of word representation, and we looked at the sequence label problem, the long term dependence, the deep learning, the machine learning, and we also studied the training methods that we used. The analytical methods and learning experiences we found from the literature review will help our project. In the next Chapter, we will introduce the methods and the algorithm that apply to this project and we will discuss them in detail.

Chapter 3 Methodology

This chapter introduces the four methods that we conceptualised in our research project. We first present an overview of each of the methods and then describe each of the methods step by step.

3.1 Introduction

Through the literature review, we have basic knowledge of NER. In the Chapter, we have mainly introduced the four methods in particular. First, we have introduced our research design. Then, we introduced the algorithm of the SVM model, the HMM model, the CRF model, and the Bi-LSTM model, respectively. We have listed the mathematical formula that we used for each step and we explain how the formulas work with NER examples specifically.

For the SVM model, we explain the data input and the concept of the SVM model and how it works, and how to classify the multi-category data via the SVM model. In relation to the HMM model, we introduce the basic idea of the HMM model, and how to find the optimal parameters of the HMM model through the given data. We also introduce the Viterbi algorithm and how to use it to find the prediction results. In regard to the CRF model, we introduce the algorithms that are used for training the CRF model and decode the method to predict the word. In terms of the Bi-LSTM model, above all, we especially describe the algorithm, and the reason to use this algorithm for each of the layers.

3.2 Research Design

Our main research content is about NER. Before the project starts, the first step is designing the research project design, and then there is the data collection, preparing the training dataset, starting the experiment, and finally getting the results of the experiment. The basic ideas of each step are necessary before the project starts.

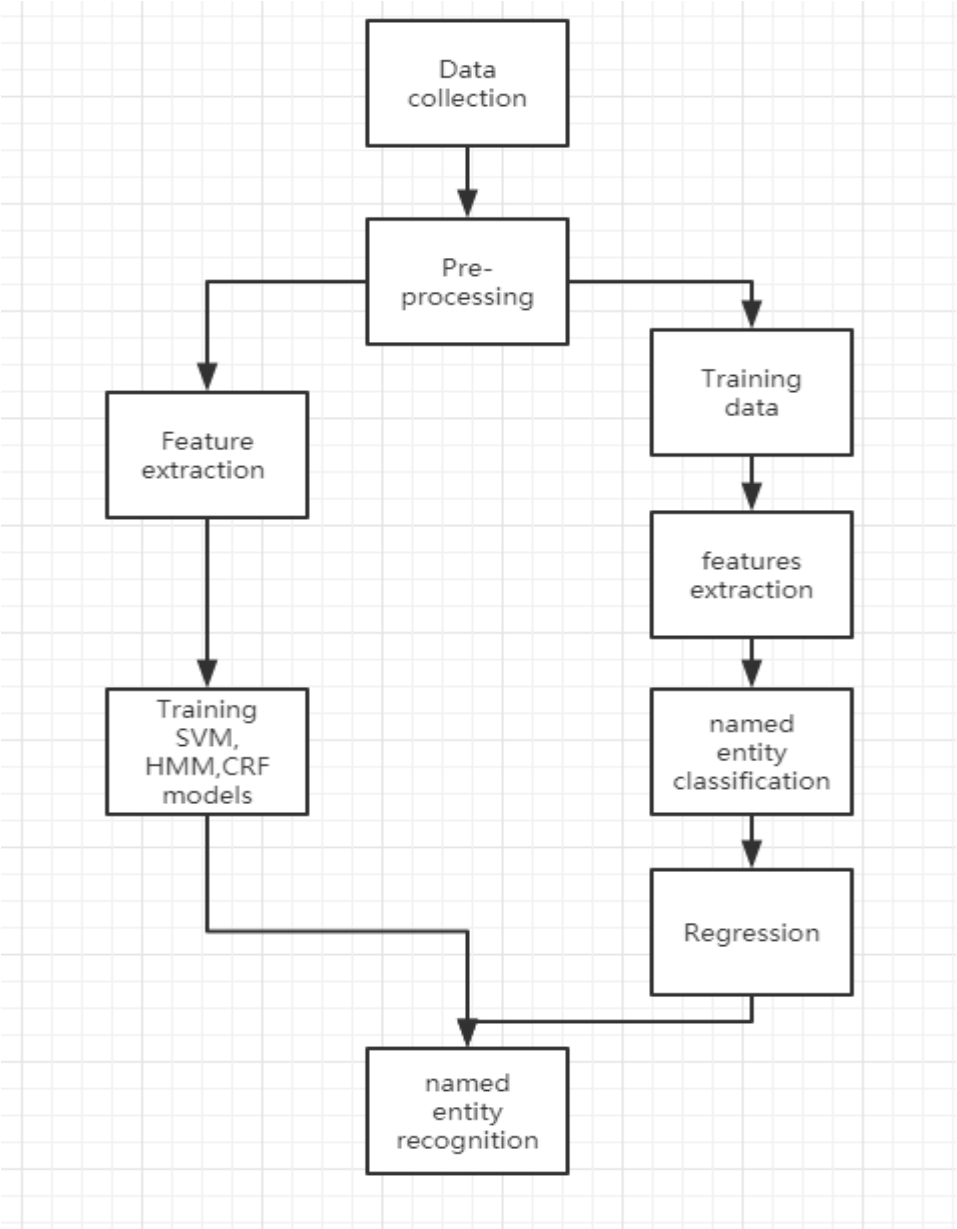

Figure 7: The steps of NER

The flow chart shown above is a summary of the particular steps involved in the research process. Regarding the research questions, the data collection is the basic step of NER. We need to find the same format database or create two databases which is the same format through the pre-processing step. After the data pre-processing, we feed data into three traditional machine learning methods and one deep learning method for NER. In the

end, we get the results and analyse the results.

3.3 SVM Model

In our project, we use LIBSVM (Chang & Lin, 2011) to train the model. The LIBSVM tool requires the following format as input data:

[label] [index1]:[value1] [index2]:[value2] ...

[label] [index1]:[value1] [index2]:[value2] ...

Figure 8: The format of LIBSVM

The label represents a category of the named entity, the index represents the feature index, and the value represents the data of a feature. We assigned 1 or 0 to every one of the named entity tags, part-of-speech tags, and to the chunk tags. We assigned "1" if the word belongs to a class or has a feature otherwise we set "0". The named entity tags, part-of-speech tags, and chunk tags are shown below in Table 1.

Named entity tags	part-of-speech tags	chunk tags
B-LOC	NN	B-NP
I-LOC	NNP	I-NP
B-PER	VB	B-VP
I-PER	IN	I-VP
B-ORG	DT	B-PP
I-ORG	NNPS	I-PP
B-MISC	CD	B-SBAR
I-MISC	VBD	I-SBAR
O	PRP	B-ADJP
	JJ	I-ADJP
	VBP	B-ADVP
	CC	I-ADVP
	VBG	B-PRT
	TO	I-PRT
	NNS	B-CONJP
	JJS	I-CONJP
	WRB	B-INTJ
	RB	I-INTJ
	WDT	B-LST
	VBN	I-LST
	POS	O

	RP	
	VBZ	
	JJR	
	WP	
	MD	
	LS	
	SYM	
	FW	
	RBS	
	EX	
	RBR	
	PDT	
	UH	
	WP$	
	PRP$	
	O	

Table 1: Table of named entity tags, part-of-speech tags, and chunk tags

According to the table above, we get 58 feature values. If a word meets any of rules, we set value "1" otherwise set "0". If a value is "0", we ignore the feature index and value. Figure shows a sample data translated to LIBSVM data format:

```
U.N.      6 2:1 2:1
official  9 1:1 2:1
Ekeus     4 2:1 2:1
heads     9 23:1 4:1
for       9 4:1 6:1
Baghdad   2 2:1 2:1
.         0 37:1 21:1
```

Figure 9: Sample of LIBSVM input data

The training of the SVM can formulate as an optimization problem (Kazama, Makino, Ohta, & Tsujii, 2002). The optimization problem can be translated to find the minimum value with a constrained condition. The optimization function is expressed as the equation.

$$min\frac{1}{2}\|w\|^2, subject.to.y_i(w\cdot x_i+b)\geq 0, i=1,\cdots,L$$

i is the total number of simple. For linear function $g(x)=wx+b$, we need to find two parameters, w (n dimension vector) and b. If we know w, we can calculate b by bringing some sample points into the equation.

Our data is not linearly separable in a two-dimensional space, thus we use the kernel function mapping data to a higher dimensional space. Therefore, the problem is expressed as the following equation:

$$f(x) = \sum_{i=1}^{m} w_{\mathrm{i}} K(x, \mathrm{z_i}) + b$$

$K(x, z_i)$ is kernel function (Ekbal & Bandyopadhyay, 2008). In practical applications, people usually choose from some commonly used kernel functions, such as, linear kernel (Leslie, Eskin, & Noble, 2002), Gaussian kernel (Keerthi & Lin, 2003), polynomial kernel (Smits & Jordaan, 2002), Laplace kernel (Gómez-Chova, Camps-Valls, Munoz-Mari, & Calpe, 2008), sigmoid kernel (Lin & Lin, 2003), and so forth.

In this project, we have chosen the linear kernel as our kernel function.

For our project, we add a slack variable $\xi_{\mathrm{i}} \geq 0$ to enable the model soft margin maximization. At the same time, each slack variable cost a penalty function C. Thus, the above equation is changed to the following:

$$min \frac{1}{2} \|w\|^2 + C \sum_{i=l}^{L} \xi_{\mathrm{i}}, \mathrm{subject.to.} \, y_i(w \cdot x_i + b) \geq 1 - \xi_{\mathrm{i}}, i = 1, \cdots, L, \xi_{\mathrm{i}} \geq 0$$

Our data has multiple classifications. We have applied the one-versus-one method (Chang & Lin, 2011) to model the models. The one-versus-one method designs an SVM model between any two types of samples. If the data has K in number of the classifications, it needs to design $k(k-1)/2$ of SVMs. If classifying an unknown sample, the method will count the results of all of the SVM models. If a category gets the most votes, the unknown sample belongs to this category. Thus, for the CoNLL 2003 dataset, we built 36 models, and for the BTC dataset, we built 21 models.

3.4 HMM Model

There are five important elements for an HMM model.

1. Q, is a collection of all of the possible states $\mathrm{Q} = \{q_1, q_2, \cdots, q_n\}$
2. V, is a discrete set of the possible observations $\mathrm{V} = \{v_1, v_2, \cdots, v_m\}$
3. A is the matrix of probability of state transition. $\mathrm{A} = \{a_{ij}\}_{N \times N}$, $a_{ij} = \mathrm{P}(q_j \, at \, t+1 | q_i \, at \, t)$ represents the probability of the translation of q_i to q_j when time t to t+1.
4. B is the observation probability distribution in state j. $\mathrm{B} = \{b_j(k)\}_{N \times M}$, $b_j(k) = \mathrm{P}(v_k \, at \, t | q_j \, at \, t)$. represents the observed value and is the probability of v_k when

time is t and the state is q_j. N is the number of possible states, M is the number of possible observations.

5. Π represents the initial state distribution. $\pi = (\pi_i)$, $\pi_i = \mathrm{P}(q_i \, at \, t = 1)$ The hidden state of the first hidden state node is the probability of each of Q, and then π is its probability distribution (Rabiner & Juang, 1986).

The relationship is as shown below:

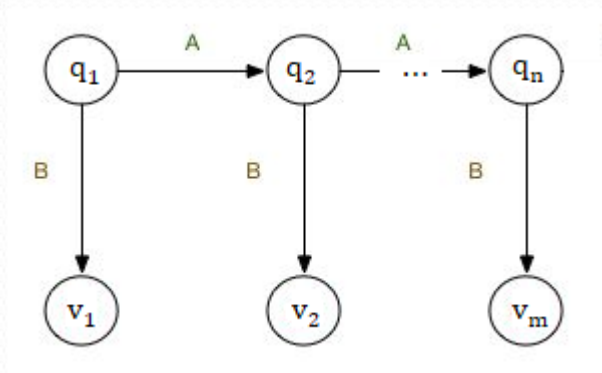

Figure 10: The relationship of elements in the HMM model

The HMM model represents the following equation (Morwal, Jahan, & Chopra, 2012)

$$\lambda = (A, B, \pi)$$

For our project, S represents words and is observable. T represents labels, we can translate it to maths representation, set a sentence as a sequence of observations, so for each word of the sentence:

$$\mathrm{S}_1^{\mathrm{n}} = \mathrm{s}_1, \mathrm{s}_2 \cdots \mathrm{s}_{\mathrm{n}}$$

The tag of each word can be regarded as corresponding to the state sequence:

$$T_1^n = t_1, t_2 \cdots t_n$$

For training data, translate to a maths representation is for a given observable state sequences T:

$$T = \{(s_1, t_1), (s_2, t_2), \cdots (s_n, t_n)\}$$

Find the optimal parameter λ of an HMM, and let $P(T|\lambda)$ maximum. We use the maximum likelihood algorithm to find the optimal parameters of an HMM. Even though the algorithm creates an initial estimation of the parameters of an HMM, this is probably an incorrect guess. Evaluating the validity of these parameters is by the given data and reducing the deviations they cause. It is necessary to keep updating the HMM model until the optimal parameters are found (Rabiner & Juang, 1986). The goal is to find a set of parameters (A and B) such that under this set of parameters, the probability of the observed data is the largest.

So, the parameters A, B and π follow equations:

$$\mathrm{A} = [a_{ij}] = \frac{A_{ij}}{\sum_{s=1}^{N} A_{is}}$$

$$\mathrm{B} = [b_j(k)] = \frac{B_{jk}}{\sum_{s=1}^{M} B_{js}}$$

$$\pi = \pi(i) = \frac{C(i)}{\sum_{s=1}^{N} C(s)}$$

A_{ij} is frequency statistics of state $t_i\ to\ t_j$; B_{jk} is frequency statistics of when state is t_j and observation is t_k; $C(i)$ is frequency statistics of initial state is t_i.

After training, we apply the Viterbi algorithm to predict classification of word. It can be considered as a given model λ and observable sequence $O = \mathrm{o}_1, \mathrm{o}_2 \cdots \mathrm{o}_\mathrm{n}$, to find the corresponding most possible state sequence $\mathrm{I} = \mathrm{i}_1, \mathrm{i}_2 \cdots \mathrm{i}_\mathrm{n}$ under the given observation sequence O. Thus, we use the Viterbi algorithm to find $P(I|O)$.

First, we initialize two states:

$$\delta_1(i) = \pi_i b_i(o_1), \mathrm{i} = 1,2 \cdots N$$

$$\psi_1(i) = 0, \mathrm{i} = 1,2 \cdots N$$

$\delta_1(i)$ is a probability of all possible state transition paths in hidden state i at time 1.

$\psi_1(i)$ is an initialization hidden state of the $t-1$th node in the most probable transition path among all of the single state transition paths whose state it is hidden at time t.

Second, estimating the local state at time $t = 2,3, \cdots T$

$$\delta_t(i) = \max_{1 \le j \le N} \left[\delta_{t-1}(j) a_{ji}\right] b_i(O_t), i = 1,2 \cdots N$$

$$\psi_t(i) = \arg \max_{1 \le j \le N} \left[\delta_{t-1}(j) a_{ji}\right], i = 1,2 \cdots N$$

Third, calculate the maximum $\delta_T(i)$ of the time T, which is the probability wherein the most likely hidden state sequence appears. Calculate the maximum $\psi_T(i)$ at time T, which is the most likely hidden state of time T:

$$\mathrm{P} = \max_{1 \le j \le N} \delta_T(i)$$

$$i_T = arg \max_{1 \le j \le N} [\delta_T(i)]$$

The last step is to use the local state $\psi(i)$ to backtracking, for $t = T-1, T-2, \cdots, 1$

$$i_t = \psi_{t+1}(i_{t+1})$$

Finally, get the most possible hidden state $I = i_1, i_2 \cdots i_\mathrm{n}$. The output I is the predict named entity sequence.

3.5 CRF Model

In our project, we assume the inputs and outputs are linear chains and $P(Y|X)$ obey the

Markov property. We need to find the parameters to build the model, so by given input word sequence $X = X_1, X_2, \cdots, X_n$ and label sequence $Y = Y_1, Y_2, \cdots, Y_n$. The corresponding equation for $P(Y|X)$ is:

$$\mathrm{P}(Y|X) = \frac{1}{Z(X)} exp\left(\sum_{i,k} \lambda_k \, t_k(Y_{i-1}, Y_i, X, i) + \sum_{i,l} \mu_l s_l \, (Y_i, X, i)\right)$$

$$\mathrm{Z(X)} = \sum_{Y} exp\left(\sum_{i,k} \lambda_k \, t_k(Y_{i-1}, Y_i, X, i) + \sum_{i,l} \mu_l s_l \, (Y_i, X, i)\right)$$

$t_k(Y_{i-1}, Y_i, X, i)$ represents where given sequence X, the probability of value translation of sequence Y at position i -1 to i. $s_l(Y_i, X, i)$ represents where given sequence X, the probability of corresponding value of sequence Y at position i. λ_k and μ_l is weights of two functions.

$t_k(Y_{i-1}, Y_i, X, i)$ and $s_l(Y_i, X, i)$ are feature functions, if we let $s_l(Y_i, X, i) = s_l(Y_{i-1}, Y_i, X, i)$ we can set them by following equation:

$$F_i = \begin{cases} 1, condition\ about\ Y_{i-1}, Y_i \\ 0\ , otherwise \end{cases}$$

Thus for a given sentence (words) X we can give a score to label sequence Y, the equation is:

$$\mathrm{score}(Y|X) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j F_j(Y'_{i-1}, Y'_i, X, i)$$

then we can find the probability of the score:

$$\mathrm{p}(Y|X) = \frac{exp[\mathrm{score}(Y|X)]}{\sum_{Y'} exp[\mathrm{score}(Y'|X)]}$$

We used the log-likelihood algorithm as an estimation function to estimate the weights of CRF. The equation is:

$$L(\lambda) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j F_j(Y'_{i-1}, Y'_i, X, i) - \sum_{j=1}^{m} \log Z_\lambda(X_n)$$

Then we use an improved iterative scaling (IIS) algorithm to find vector increment δ by following equations:

$$E_{p'}[t_k] = \sum_{x,y} P'(x, y) \sum_{i=1}^{n+1} t_k(Y_{i-1}, Y_i, X, i)$$

$$E_{p'}[s_l] = \sum_{x,y} P'(x, y) \sum_{i=1}^{n+1} s_l(Y_i, X, i)$$

And then use δ to update current parameter $\lambda = \lambda + \delta$. If not all λ is convergence, repeat it.

We know feature vectors and weight vectors and observed sequences, then we use the Viterbi algorithm to predict the classification of a word. The first step is initialization by

the equation:

$$\delta_1(j) = wF_1(y_0 = start, y_1 = j, x), j = 1,2 \cdots m$$

Then find maximum of probability of each label l at position i

$$\delta_t(l) = \max_{1 \le j \le m} [\delta_{i-1}(j) + wF_1(y_{i-1} = j, y_i = l, x)], l = 1,2 \cdots m$$

Record the paths.

$$\psi_t(l) = arg \max_{1 \le j \le m} [\delta_{i-1}(j) + wF_1(y_{i-1} = j, y_i = l, x)], l = 1,2 \cdots m$$

Stop at $i = n$, at this position, the maximum probability is:

$$\max_y (wF(y, x)) = \max_{1 \le j \le m} \delta_n(j)$$

End of the optimal path:

$$y'_n = arg \max_{1 \le j \le m} \delta_n(j)$$

From the optimal path go back:

$$y'_i = \psi_{t+1}(y'_{i+1}), \mathrm{i} = \mathrm{n} - 1, \mathrm{n} - 2, \cdots, 1$$

In the end, we get optimal path $y' = (y'_1, y'_2, \cdots, y'_n)^T$. The output path y' is the predict named entity sequence.

3.6 Bi-LSTM-CNN-CRF Model

We chose Bi-LSTM-CNN-CRF to be the deep learning method for NER. Bi-LSTM-CNN-CRF is a typical deep learning method. The features will be extracted through the embedding layer, then fed into the LSTM network, the last step is to classify through the CRF layer. As shown in Figure 11, the diagram is the basic architecture of the Bi-LSTM-CNN-CRF network.

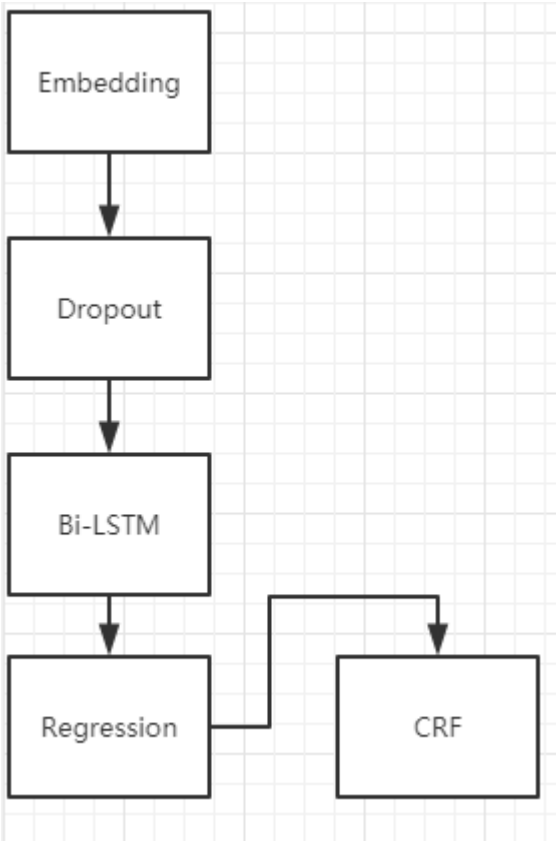

Figure 11: Architecture of the Bi-LSTM-CNN-CRF network

- Embedding layer

The first layer is the embedding layers, which contains two sublayers, word-level embedding and char-level embedding. For word-level embedding, we used GloVe Twitter vectors set as pre-trained word vectors. There are 37.17% of Out of vocabulary (OOV). For char-level embedding, we used CNN to extract the character-level representation of the words. Many previous research studies proved that extracting the features from the characters of a word through CNN is an effective method (Santos & Zadrozny, 2014) (Chiu & Nichols, 2016).

- Dropout layer

In order to reduce the overfitting of the embedding network, we used the dropout strategy. It randomly disables some of nodes. According to Ma and Hovy (Ma & Hovy, 2016), we add dropout on char embedding before feeding into CNN. Also, the dropout layer is used to regularise the input and output vectors of the Bi-LSTM model. During the model forward propagation, let the activation value of some neuron stop working with a certain probability P, which makes the model more generalized because it does not depend too much on some local features. The dropout function followed Equations:

$$r_j^{(l)} \sim Bernoulli(p)$$

$$\acute{y}^{(l)} = y^{(l)} \times r^{(l)}$$

$r_j^{(l)}$ It is a vector consisting of 0 and 1. Each value of r is chosen from a Bernoulli distribution by probability P. $\gamma^{(l)}$ is the output value of activation function. Thus, a node stop work means its output value of activation function multiply by 0 in the vector $r_j^{(l)}$. The probability of a node stop work is P.

When model testing each node needs to be multiplied by P.

$$\omega^{(l)} = pW^{(l)}$$

In our model, we set P is 0.5, which means every node has half of a probability to stop working.

- Bi-LSTM

The input of Bi-LSTM network involves combining the word-level embedding vectors and char-level embedding vectors as features together. Thus, we use each combination embedding sequence $(x_1, x_2, \cdots, x_n)$ as each time step input of the Bi-LSTM network. Through processing and calculation, gain hidden states sequence. Each layer computes the following functions (Graves, 2013):

$$i_t = \sigma(W_{hi}h_{t-1} + b_{hi} + U_{ii}x_t + b_{ii})$$

$$f_t = \sigma\big(W_{hf}h_{t-1} + b_{hf} + U_{if}x_t + b_{if}\big)$$

$$g_t = \tanh\big(W_{hg}h_{t-1} + b_{hg} + U_{ig}x_t + b_{ig}\big)$$

$$o_t = \sigma(W_{ho}h_{t-1} + b_{ho} + U_{io}x_t + b_{io})$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

$$h_t = o_t * \tanh(c_t)$$

Where x_t is the input at time t. i_t, f_t, g_t and o_t represent input, forget, cell and output gates. h_t is the hidden layer at time t, c_t is cell state at time t, h_{t-1} is the hidden state of the layer at time $t - b1$ or initial hidden state at time 0. U_{ii}, U_{if}, U_{io} and U_{ig} are weight for input to hidden state of input gate, forget gate, output gate and cell state. W_{hf}, W_{hi}, W_{hg} and W_{ho} are weight for hidden state of forget gate, input gate, cell state, output gate to hidden layer. b_{if}, b_{ii}, b_{ig} and b_{io} are input-hidden bias. b_{hi}, b_{hf}, b_{hg} and b_{ho} are hidden-hidden bias. σ represents sigmoid function and $*$ is the Hadamard product. All weights and biases are initialized from range $\mu\big(-\sqrt{k}, \sqrt{k}\big)$ where:

$$k = \frac{1}{h}$$

h is the number of features in the hidden state.

In the input gate, the forget gate and the output gate, we use the Sigmoid algorithm as the activation function for gates, which could be represented as the equation:

$$S(t) = \frac{1}{1 + e^{-t}}$$

To reduce overfitting, we add the dropout function to the Bi-LSTM and use TanHyperbolic (tanh) to activate the hidden layer of Bi-LSTM. It follows the equation:

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

After this, combine the forward LSTM output $\left(\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_n}\right)$ and the backward LSTM output $\left(\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots, \overleftarrow{h_n}\right)$ in correspond position $h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$. The hidden state sequence belongs to integer set:

$$(h_1, h_2, \cdots, h_n) \in R^{n \times m}$$

- Linear layer

Before putting hidden states sequence into CRF layer, we need to transform from m dimension to k dimension of hidden states sequence. K is total number of classifications.

It gets extracted features from a sentence as a matrix $P_i \in R^k$. Each p_{ij} from P_i can be regarded as a score that x_i is classified to j-th classification.

- CRF layer

The CRF layer performs sentence-level sequence labelling. The parameter of the CRF layer is a $(k+2) \times (k+2)$ matrix A. A_{ij} represents a score from label i to transform to label j. The reason for plus 2 is because adding two extra states into label sequence. A starting state represents the beginning of a sentence and a termination state represents the end of a sentence.

For a label sequence $y = (y_1, y_2, \cdots, y_n)$, the score that giving by CRF layer is:

$$\text{score}(x, y) = \sum_{i=1}^{n} P_{i,y_i} + \sum_{i=1}^{n+1} A_{y_{i-1} y_i}$$

So, the scoring of the entire sequence is equal to the sum of the scores of the respective positions. The scoring of each position is obtained by two parts, one part is determined by the P_i of the output of Bi-LSTM layer, and the other part is determined by the transfer matrix A of the CRF.

Then get the probability after normalized through Softmax:

$$P(y|x) = \frac{exp\big(score(x, y)\big)}{\sum_{y'} exp\big(score(x, y')\big)}$$

Max log likelihood is generally used as a loss function. The model classification loss function is the following equation:

$$\log P(y|x) = score(x, y) - \log\left(\sum_{y'} exp\big(score(x, y')\big)\right)$$

3.7 Summary

This section introduces how to process the data for the different models. Training the SVM model, the HMM model, the CRF model, and the Bi-LSTM-CNN-CRF model is part of our main contribution. Through this Chapter, we clearly understand the methods and algorithms. In the next Chapter, we will show the results of our experiment.

Chapter 4 Results and Analysis

In this chapter, we display our implementation of named entity recognition and the methods and the results that we got. Firstly, we introduced two corpora and data pre-processing. Then we showed the hardware and the software experimental environments. In the end, we evaluated the results of the four methods.

4.1 Introduction

We got the results from the four methods that we used respectively, for NER through the experiments. This Chapter focuses on the outcomes that we collected from the four methods. We use the confusion matrix and the classification report to evaluate each method. At first, the two corpora used in this project will be introduced in detail. From the aspect of word representation, we need to carry out the pre-processing of the data that apply for the different models. This is to ensure that the models can extract enough features from the word representation. Then, we introduced our experimental environment for this project. In this part, we describe each software and library in detail. We also give the hardware information that we use to accelerate the training. Then we display the results of each method. Each method trains two models through two different corpora. We present a detailed analysis of the performance of each method in two databases with visual charts. We list and analyse the result from every class of each model and then compare the performance of the four methods that work on each corpus. At the end, we discuss the limitations of our experiment.

4.2 Data Description

In this project, we used two datasets, the first one from the CoNLL 2003 shared task, the second from the Broad Twitter Corpus.

4.2.1 CoNLL 2003

CoNLL 2003 shared task is a very famous NER database. It includes two languages, English and German. The English data was collected from newswire articles by Reuters Corpus, and by people from the University of Antwerp who did the annotation for the data. The data used Reuters news stories from August 1996 to August 1997. It consists of three files, training set, development set, and test set. The data of the training set and the development set started from the end of August 1996. The data of the test set began in December 1996. The pre-processing data also included some data from September 1996. The German data was collected from the ECI Multilingual Text Corpus3. The data in CoNLL 2003 shared task was collected from the German newspaper Frankfurter Rundschau. All three datasets were extracted from articles written during one week in August 1992. The original data started between September to December in 1992.

English data	Articles	Sentences	Tokens
Training set	946	14987	203621
Development set	216	3466	51362
Test set	231	3684	46435

Table 2: Number of articles, sentences and tokens in each data file of the English data set

German data	Articles	Sentences	Tokens
Training set	553	12705	206931
Development set	201	3068	51444
Test set	155	3160	51943

Table 3: Number of articles, sentences and tokens in each data file of the German data set

Table 2 and 3 shown above contain the size of each file of two datasets.
All of the data followed the same format. Each sentence was segmented into words, punctuation and so on and set each word to a separate line. The empty line represented the sentence boundaries. Each line contains a word, part-of-speech, chunk tag and tag of the named entity, each field was split by space. An example is shown below.

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Figure 12: Sample of the CoNLL 2003 dataset

The data contains four types of named entities, there is the person represented as PER, organizations are represented as ORG, locations are represented as LOC, and miscellaneous names are represented as MISC. We also used the Ramshaw and Marcus proposed IOB tagging scheme to represent the boundary of the named entities. Tag O represents the outside of the named entities, I mains the inside of the named entities, B represents the beginning word of the second named entity, which is immediately near to the first named entity. Table 4 and 5 shown below contain the specific numbers of the named entities in each file of the two datasets.

English data	LOC	MISC	ORG	PER
Training set	7140	3438	6321	6600
Development set	1837	922	1341	1842
Test set	1668	702	1661	1617

Table 4: Number of named entities per data file of the English data set

German data	LOC	MISC	ORG	PER
Training set	4363	2288	2427	2773
Development set	1181	1010	1241	1401
Test set	1035	670	773	1195

Table 5: Number of named entities per data file of the German data set

4.2.2 Broad Twitter Corpus

Broad Twitter Corpus is the full open NER dataset. It was collected from Twitter posts over six years, including the reaction of news stories, non-professional content, and tweets from Twitter. In order to represent the maximum diversity, the data has been extracted from different periods, such as, different months of the year, different days of the month, and different times of the day. In addition, the data came from different English-speaking countries and regions. In order to reflect the diversity and used different annotation approach, the corpus was organized into six segments. The corpus was

annotated by eight NLP experts and 747 crowds of workers. The data used the PLO entity schema and IOB tagging scheme, PLO as represents the named entity types of person, location, and organization. The final corpus contains 12K named entities, 5271 of person entities, 3114 location entities, and 3732 of organization entities in total. The Broad Twitter Corpus did not label part-of-speech and chunk tag. An example is shown above.

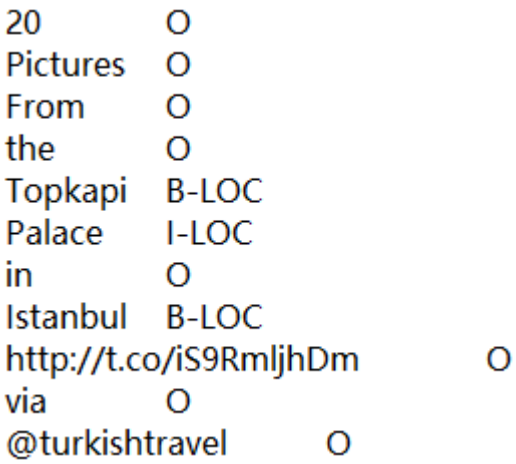

Figure 13: Sample of Broad Twitter Corpus

Segment A contains the data collected from UK tweets near the New Year. It is annotated by different NLP experts.

Segment B same like Segment A, it contains more non-directed tweets. It is annotated by different NLP experts.

Segment E contains tweets that are commentary from different places related to the flight crash of Malaysia Airlines flight MH17. It contains some comments that come from non-native speakers, usually from Ukrainian, Dutch or Malaysian speakers. It is annotated by different NLP experts and crowd.

Segment F contains popular personal content from Twitterati. These authors come from six English-speaking regions, and the commentary is related to celebrity, music, politics, sports and news. It is annotated by different NLP experts and crowds.

Segment G comprises tweets from mainstream news in six English-speaking regions.

Segment H excludes tweets from the UK to balance the UK bias of Segment B. It is also collected from different periods. It is annotated by different NLP experts and crowds.

Table 6 shows the difference of the six segments.

Section	Region	Collection period	description	annotators	Tweets
A	UK	2012.01	General collection	Expert	1000
B	UK	2012.01-02	Non-directed tweets	Expert	2000
E	Global	2014.07	Related to	Expert &	200

			MH17 disaster	Crowd	
F	Stratified	2009-2014	Twitterati	Expert & Crowd	2000
G	Stratified	2011-2014	Mainstream News	Expert & Crowd	2351
H	Non-UK	2014	General collection	Expert & Crowd	2000

Table 6: Sections of corpus A region of "stratified" indicates that data was taken from six regions in the English-speaking world

4.3 Pre-processing of Dataset

After the data collection, the CoNLL 2003 data do not require pre-processing, it is already tagged and chunked by the memory-based MBT tagger and labelled by the University of Antwerp.

For the BTC dataset, I randomly chose 80% of the sentences from section A, B, E, F, and G as the training set and the rest of those as the development set. Section H is the testing set. The reason is because section H removes the tweets from the UK origin, it avoids the bias of the UK, so it the most suitable dataset for a test. Also, the sentence order of the training data is randomly taken from section A, B, E, F, G. Table 7 below contains the particular numbers of the named entities of the three datasets.

English data	PER	ORG	LOC
Training set	2160	1162	859
Development set	816	603	501
Test set	1372	537	297
Total	4348	2302	1657

Table 7: Number of named entities of dataset

The BTC dataset only labelled the entity tag of the data, for the SVM, the HMM, and the CRF method, part-of-speech tags and chunk tags are necessary features. Therefore, I used spaCy tool to add part-of-speech tags and chunk tags to BTC dataset. SpaCy is free open-source library. It is able to add part-of-speech tags and chunk tags through the pre-training model. I chose the en_core_web_lg model from the spaCy provided models, which is a statistical model with the highest performance. The model is a multi-task CNN based model. It trained on OntoNotes 5 corpus with GloVe vectors. It comprises 685k keys and 685k unique vectors and 300 dimensions per vector. The accuracy of POS is 96.98%.

After adding part-of-speech tags, I added chunk tags to the BTC database. The spaCy

only provides the noun chunks function that merges the noun chunks into a single token. Thus, I wrote the rest of the part of speech chunk tags function that adds the chunk tags with the IOB tagging scheme. The Verb tags function, Adjective tags function, Adverb tags function and Proposition tags function are based on part-of-speech tags, add VP, ADJ, ADV, and PP tags to the corresponding word with IOB tagging scheme. Conjunction, number, website, and hashtag, and so on are classified as others which are represented as O. After pre-processing, an example of the BTC database is shown below.

```
20 CD B-NP O
Pictures NNS I-NP O
From IN B-PP O
the DT B-NP O
Topkapi NNP I-NP B-LOC
Palace NNP I-NP I-LOC
in IN B-PP O
Istanbul NNP B-NP B-LOC
http://t.co/is9RmljhDm NFP O O
via IN B-PP O
@turkishtravel ADD O O
```

Figure 14: Sample of Broad Twitter Corpus after pre-processing

4.4 Experimental Environment

- Hardware experimental environment:

The experiment is run on a desktop with 4 core i5 CPU 3.0GHZ and NVIDIA GeForce GTX 1060 6GB GPU. We used GUP to accelerate the neural network training and testing.

- Software experimental environment:

We used Python as the programming language for the training and testing of the models. Python 2.7 is used for modelling machine learning based methods. For the SVM model we used the LIBSVM library (Chang & Lin, 2011);

The particular software version we used is shown below:

Software name and version	Function
Python 2.7	HMM modelling, CRF modelling
spaCy 2.1.3	Data pre-processing
LIBSVM 3.23	SVM modelling

Table 8: Software version of machine learning methods

For the Bi-LSTM-CNN-CRF (Ma & Hovy, 2016) model we built it through the PyTorch platform, and we used the GUP to accelerate the model training and testing. We used

Matplotlib and Sklearn as our analysis and visualization library.

Table 9 shows the software version we used for the Bi-LSTM-CNN-CRF model.

Software name and version	Function
Python 3.6	Modelling environment
PyTorch 1.0.1	Bi-LSTM-CNN-CRF modelling
CUDA 9.1	GPU acceleration
cuDNN 7.1.3	GPU-accelerated library

Table 9: Software version of Bi-LSTM-CNN-CRF model

4.5 NER Results

4.5.1 Overall Results

Four methods are introduced in Chapter 3. We used four methods to model the two corpora separately. The results of the testing set are shown in Table 10 below.

	CoNLL2003			BTC		
	Precision	Recall	F1	Precision	Recall	F1
SVM	72.53%	54.32%	62.12%	56.71%	44.57%	49.91%
HMM	85.04%	67.75%	74.57%	65.68%	49.42%	56.40%
CRF	83.72%	79.03%	80.94%	70.14%	54.86%	61.57%
Bi-LSTM	89.73%	90.21%	89.97%	76.57%	70.14%	73.22%

Table 10: Results of our experiment

We used precision, recall, and F1 value to evaluate the performance of the four methods. For the CoNLL2003 corpus, the Bi-LSTM is the best of the four models. The F1 value of Bi-LSTM is 89.97% from the testing set. For the BTC corpus, the Bi-LSTM model still got the best performance in the four models. The SVM method got the worst performance of the four methods. Comparing the overall results, the performance of the four methods in the CoNLL2003 corpus are better than their performances in the BTC corpus. This shows that the performance of the informal data that used the same method still needs to be improved. The noisy, informal, and unstructured data has a significant impact on the performance of the existing methods.

In order to further analyse and compare the four models, we used the visual confusion matrix and the classification report to display the predicted results of the models. The visual confusion matrix uses different coloured squares to represent the number, and it can see where the prediction error is clearly because all of the correct predictions are on the diagonal and the prediction error is outside the diagonal. The classification report lists the precision, recall, and F1 score of each class, also calculates the overall Micro

average, Macro average, and Weighted average of the precision, recall, and F1. The three average values will help us better measure the performance of the models. The macro average gives each class the same weight, whereas the micro-average provides each sample with the same weight. Most of the classes of the corpus are none entity. Taking into account the sample skew, the Macro average is the best value to measure the models (Özgür, Özgür & Güngör, 2005). The macro average is the arithmetic mean of the precision, recall, and F1 value for all categories. The Macro-precision, Macro- recall and Macro- F1 score are expressed as shown below (Sokolova & Lapalme, 2009):

$$P_{macro} = \frac{1}{n}\sum_{i=1}^{n} P_i$$

$$R_{macro} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$F_{macro} = \frac{2 \times P_{macro} \times R_{macro}}{P_{macro} \times R_{macro}}$$

4.5.2 CoNLL 2003 Results

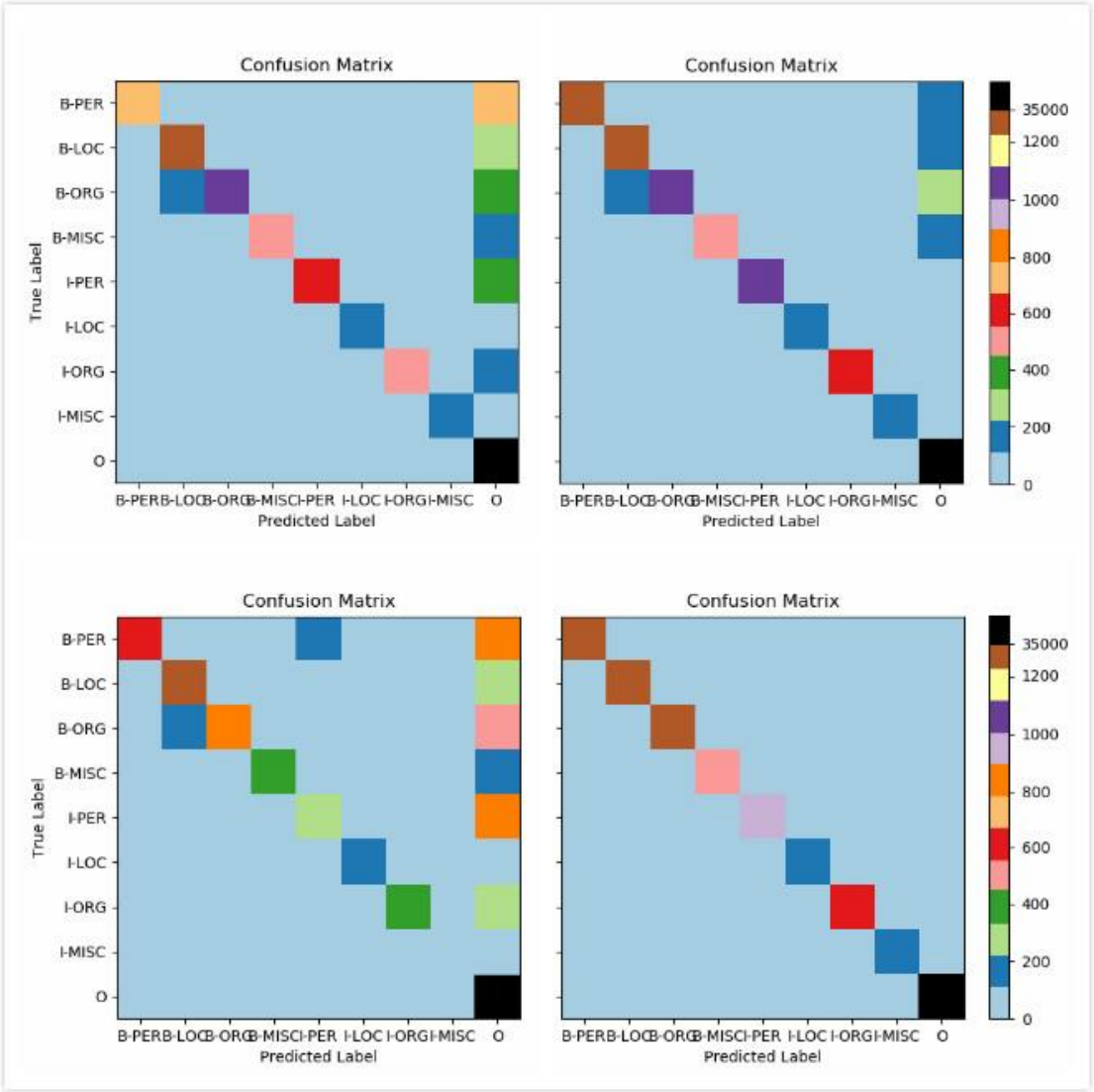

Figure 15: Confusion matrices of CoNLL 2003 result

- From Figure 15, we can see the confusion matrix of the four models. The left top of Figure 4.1 is the confusion matrix of the HMM model. In this confusion matrix, we can see that the right side of the figure is not light blue which means the prediction of the entity categories are incorrectly assigned to the non-entity categories. The prediction between the entities is quite good, there are not many entity classes that are incorrectly assigned to another entity class except for the B-ORG class. For the B-ORG class, some are predicted as the B-LOC class.
- The right top of Figure 15 is the confusion matrix for the CRF model. The entity classes are predicted as non-entity and this is the main problem of the B-BER, B-LOC, B-ORG and the B-MISC classes. For the B-ORG class, some are predicted as B-LOC class. Compared with the HMM model, the recognition of the person's name

has been significantly improved, and the I-PER class that is predicted to be non-entity is decreased. The prediction of B-PER is also improved. The identification of the location name has not progressed much, the number of B-LOC class that has been predicted to be non-entities has decreased. The name of the organization has also improved significantly. The number of I-ORG class that has been predicted to be correct has increased significantly. The MISC class has not changed much.

- The left bottom of Figure 15 is the confusion matrix for the SVM model. Some B-LOC, B-MISC, I-PER, I-ORG are identified as non-entity. The B-ORG class is predicted as the B-LOC class and the O class. The I-MISC got the worst performance. Compared with the HMM model and the CRF model, and the SVM model has a critical defect in the identification of the personal name. There are a large number of PER categories that are predicted to be non-entity categories, even exceeding the correct number of predictions.
- From the right bottom of Figure 15, we can see that confusion matrix for Bi-LSTM model. All categories showed clearly on the diagonal. The wrong predictions of Bi-LSTM model are fewer than 100. Thus, the model gets the best performance in terms of the improved predictive performance of all of the categories compared to the other three models.

	HMM model			CRF mdoel			SVM model			Bi-LSTM model			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Total number
B-PER	0.93	0.48	0.63	0.91	0.87	0.89	0.87	0.37	0.51	0.97	0.95	0.96	1617
B-LOC	0.85	0.81	0.83	0.88	0.82	0.85	0.78	0.74	0.76	0.92	0.94	0.93	1668
B-ORG	0.82	0.61	0.7	0.89	0.66	0.76	0.76	0.49	0.59	0.91	0.89	0.9	1661
B-MISC	0.82	0.73	0.77	0.87	0.72	0.79	0.74	0.55	0.63	0.82	0.84	0.83	702
I-PER	0.97	0.58	0.72	0.86	0.96	0.91	0.53	0.22	0.31	0.98	0.99	0.98	1156
I-LOC	0.78	0.66	0.72	0.66	0.76	0.71	0.53	0.47	0.5	0.82	0.91	0.86	257
I-ORG	0.84	0.61	0.7	0.85	0.7	0.77	0.54	0.44	0.48	0.87	0.87	0.87	835
I-MISC	0.71	0.63	0.67	0.64	0.63	0.64	0.25	0.31	0.28	0.71	0.73	0.72	216
O	0.94	0.99	0.97	0.98	0.99	0.98	0.93	0.99	0.96	0.99	0.99	0.99	38554
Micro Avg	0.93	0.94	0.93	0.96	0.96	0.96	0.9	0.9	0.9	0.98	0.98	0.98	46666
Macro Avg	0.85	0.68	0.75	0.84	0.79	0.81	0.66	0.51	0.56	0.89	0.9	0.89	46666
Weighted Avg	0.93	0.93	0.92	0.96	0.96	0.96	0.89	0.9	0.89	0.98	0.98	0.98	46666

Figure 16: The classification report of CoNLL 2003 result

Figure 16 shows the classification report of the four models, it lists the precision, recall and F1 score for each class. The classification report of the HMM model showed the F1 score of the B-PER class is the lowest in all of the classes. The HMM model has problems with the identification of name of the person. Since the O class accounts for the majority of the data set, the micro-average does not evaluate the overall model very well. Macro averaging is the main indicator of our evaluation model.

In the report of the CRF model, we can see that almost of the classes get over 70% of the F1 score, excluding the I-MISC. The overall F1 of the CRF model is the highest in the machine learning models. In the SVM model, the F1 of the I-MISC and the I-PER less

than 40%, the overall performance is lower than the HMM model and the CRF model. The recall of B-PER and I-PER are lower than 40 percent. Thus, the SVM model cannot be applied to person name recognition.

We can see that all of the values of F1 are greater than 70% in the Bi-LSTM model. Excluding the MISC class, the other classes got over 85% of F1. The overall performance is better than the machine learning methods.

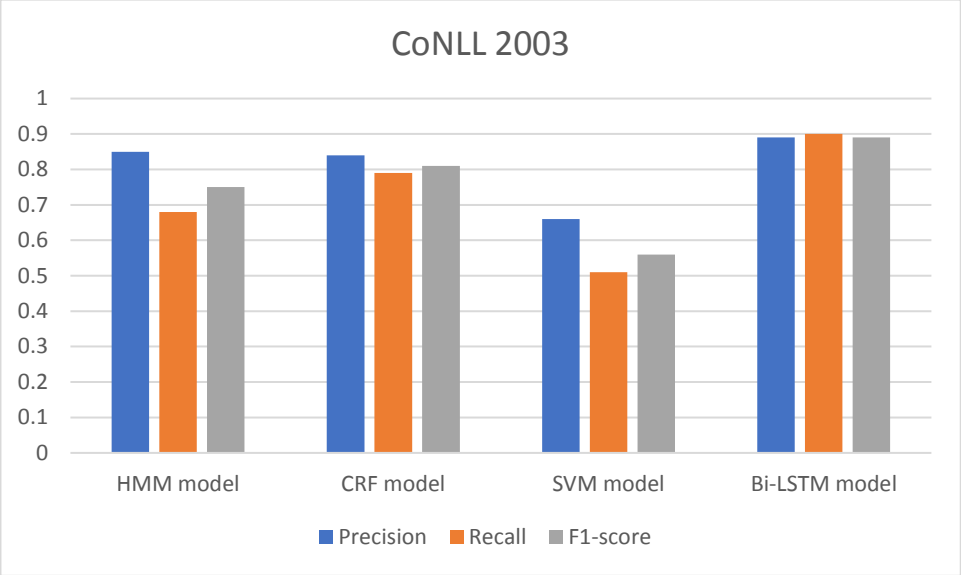

Figure 17: The overall performance of CoNLL 2003 result

Figure 17 shows the overall performance of the four models. The precision of the HMM model, the CRF model, and the Bi-LSTM model are over 80 percent. However, the recall of the HMM model is lower than the CRF model and the Bi-LSTM model. For the HMM model, only 68% of the examples are identified correctly. It is lower than the CRF model and the Bi-LSTM model. For the SVM model, many of the named entities are not detected, and many of the detected entities are incorrectly classified.

4.5.3 BTC Results

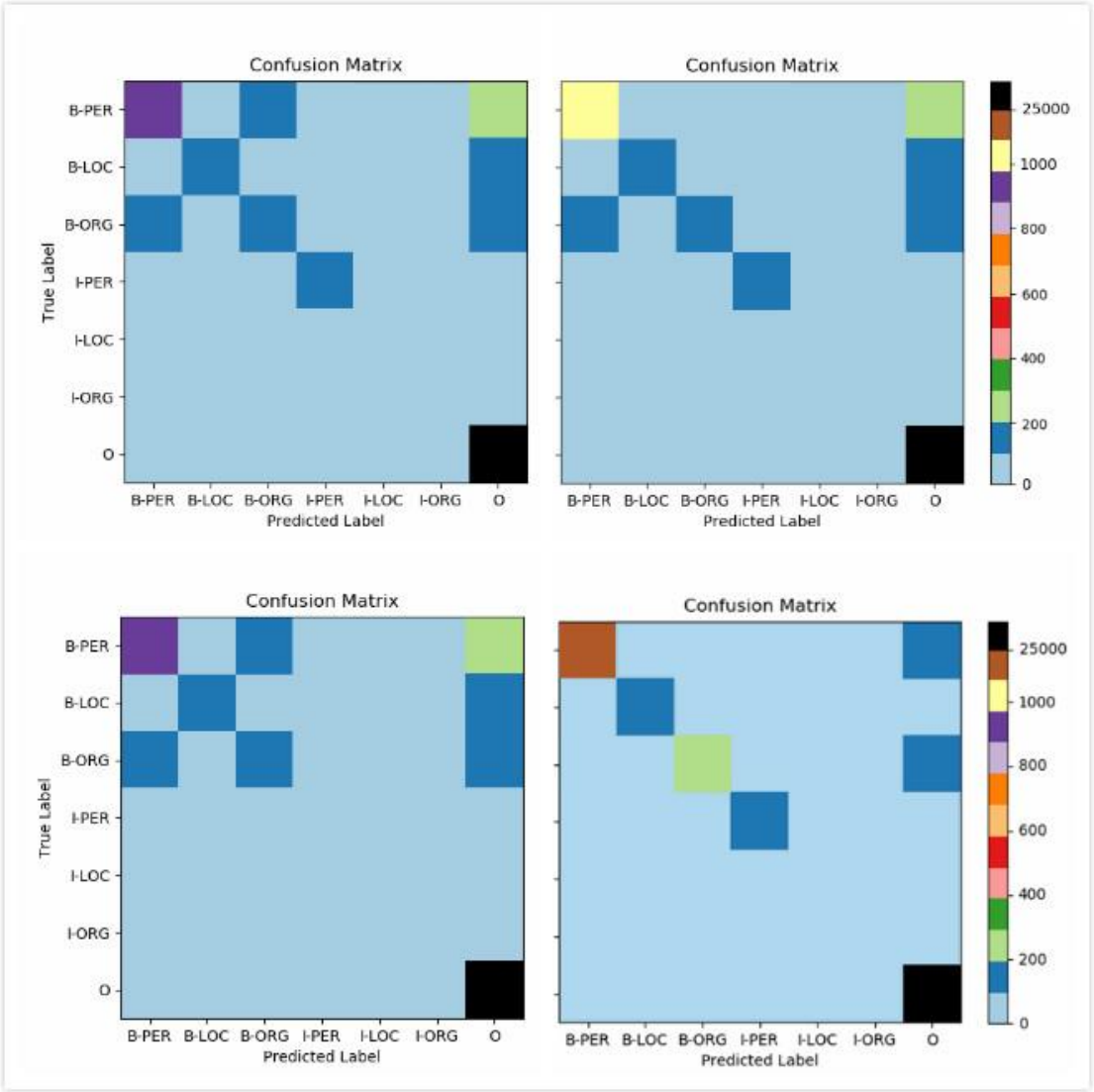

Figure 18: Confusion matrices of BTC result

- Figure 18 shows the confusion matrix of the testing set of BTC corpus. The confusion matrix of the HMM model is shown in the subfigure at the top left of Figure 4.6. The B-LOC, B-ORG, I-LOC, and I-ORG classes are not well identified. The testing set has small quantities of I-LOC and I-ORG categories. Thus, these two categories are not represented in the confusion matrix. From the model it is hard to distinguish the B-PER and the B-ORG.
- The subfigure at the top right corner of Figure 18 shows the confusion matrix of the CRF model. Compared with the HMM model, the CRF model got a better performance with respect to predicting the B-PER class. However, the B-LOC and the B-ORG classes still face the problem which is predicted as non-entity. One-third of the B-ORG class is predicted as the B-PER class. I-LOC and I-ORG classes cannot

be displayed in the confusion matrix because the number of I-LOC and I-ORG classes is too small.

- The confusion matrix of the SVM model is shown in the subfigure at the bottom left of Figure 18. Compared with the HMM model and the CRF model, the informal corpus has the most significant impact on the SVM model. The I-PER, I-LOC, and I-ORG cannot be displayed in the confusion matrix. The performance of B-PER, B-LOC, and B-ORG are also further reduced.
- The subfigure at the bottom right of the corner of Figure 18 is the confusion matrix of the Bi-LSTM model. The overall performance has been improved. The I-PER is clearly displayed on the confusion matrix. Compared with the other three models, the prediction of B-PER, B-LOC and B-ORG classes have improved significantly.

	HMM model			CRF mdoel			SVM model			Bi-LSTM model			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Total number
B-PER	0.76	0.72	0.74	0.81	0.82	0.81	0.72	0.69	0.81	0.86	0.87	0.86	1372
B-LOC	0.68	0.49	0.57	0.72	0.53	0.61	0.53	0.4	0.61	0.75	0.68	0.71	297
B-ORG	0.55	0.21	0.3	0.59	0.27	0.37	0.45	0.2	0.37	0.72	0.63	0.67	537
I-PER	0.71	0.58	0.63	0.74	0.65	0.69	0.56	0.48	0.69	0.78	0.7	0.73	196
I-LOC	0.5	0.33	0.4	0.55	0.44	0.48	0.42	0.28	0.48	0.66	0.61	0.63	114
I-ORG	0.48	0.12	0.19	0.53	0.14	0.22	0.39	0.1	0.22	0.61	0.43	0.5	127
O	0.9	0.99	0.94	0.97	0.99	0.97	0.9	0.97	0.97	0.98	0.99	0.98	26826
Micro Avg	0.88	0.88	0.88	0.95	0.95	0.95	0.87	0.87	0.87	0.96	0.96	0.96	29469
Macro Avg	0.66	0.49	0.56	0.7	0.55	0.62	0.57	0.45	0.5	0.77	0.7	0.73	29469
Weighted Avg	0.88	0.88	0.88	0.95	0.95	0.95	0.87	0.87	0.87	0.96	0.96	0.96	29469

Figure 19: The classification report of BTC result

Figure 19 shows the classification report of the BTC corpus. In the HMM model, the recall of B-ORG and I-ORG classes is very low. The correct rate of organization names that this model predicted is extremely low. The overall recall of this model is only 49% and the overall precision of this model is only 66 percent. Thus, the informal data has a huge effect on the HMM model.

Compared with the HMM model, the overall performance of the CRF model has increased slightly. However, organization name and location name still cannot be identified very well. The recognition of person name has been greatly improved. The overall performance is still not satisfactory.

The overall performance of the SVM model is the lowest model in the three machine learning methods. It almost completely fails to correctly identify all of the inside of the named entities categories.

The overall performance of the BI-LSTM model is the best model in our project. It is much higher compared to the others. However, compared with the news stories dataset, the informal dataset still has an effect on the performance of the neural model.

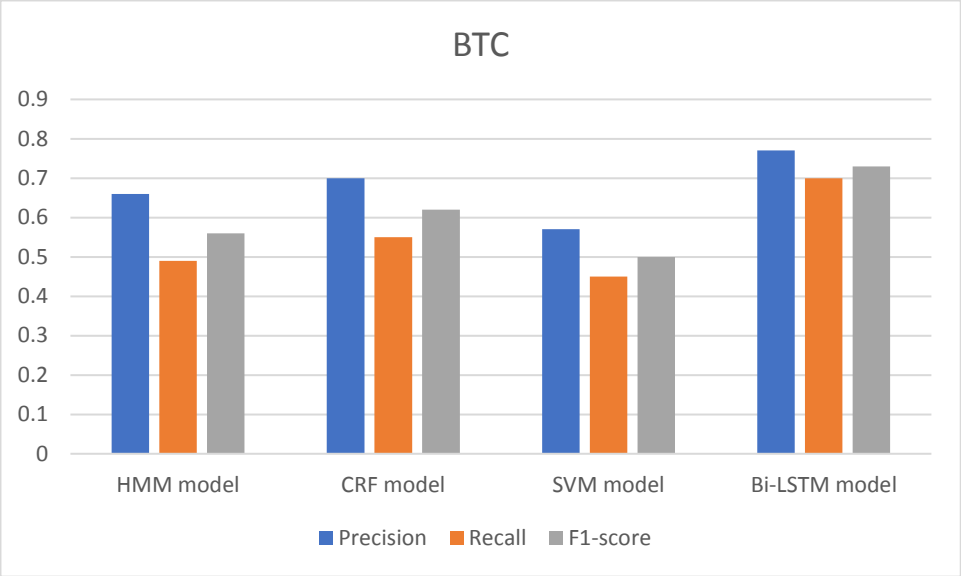

Figure 20: The overall performance of BTC result

Figure 20 shows the overall precision, recall and F1 score of the four models. The overall recall of the HMM model, the CRF model, and SVM model is lower than 55 percent. Therefore, nearly half of the three models identified entities that are incorrect, especially in the data sets where the entity class is a minority. The Bi-LSTM model is much higher than the other three models. The overall performance of the Bi-LSTM model is satisfactory.

4.4 Limitation of the Experiments

For this project, we used the SVM model, the HMM model, the CRF model, and the mixed model. The mixed model is the Bi-LSTM-CNN-CRF model. The CNN as the char-level feature extractor, the Bi-LSTM as the basic model of the overall framework, the CRF focuses on the sequence label prediction. Machine learning based methods and mixed model have their limitations. The limitations will have effects on the results of NER.

1) The BTC corpus is much smaller than the CoNLL2003. It may not fully train the mixed model. We need to find a large twitter corpus.

2) Regarding the BTC corpus, it does not include part-of-speech and chunk tags. These tags are very important features for machine learning methods. Thus, we used the spaCy library to add these features. This is because there could be mistakes in the part-of-speech and chunk tags. This is because the spaCy library cannot guarantee a 100% correct rate.

4.5 Summary

In this Chapter, we showed the results of our models. We used four methods to build models for the two corpora separately. In order to select the optimal model, each model is evaluated by precision, recall, and F1. Through the confusion matrices and classification reports, we analysed each classification for the models. We not only compared the differences between the four methods we also compared the differences between the same method used in the different corpora. In the next Chapter, we will keep analysing the results of our models, and discuss the advantages and the disadvantages of each of the models.

Chapter 5 Discussions

The results of the four methods are analysed and discussed in this Chapter. There is a special focus on the Bi-LSTM model, we analysed the influence of the different parameters on the model. We also compared the four models, and we discussed the advantages and disadvantages of the models.

5.1 Introduction

In this Chapter, we discuss the Bi-LSTM in detail, we focus on parameter tuning, and we give the performance evidence of the different values of the parameters. We discuss the fully trained aspect in the neural network model and avoid overfitting. We compare the different methods of dealing with out of vocabulary. The different initialization assignments are also discussed, showing the particular initialization assignments of our CNN network and Bi-LSTM network. We display the processing of the select optimal hyperparameters. In addition, we discuss the characteristics, the advantages, and the disadvantages of the four methods.

5.2 Analysis of Bi-LSTM Model

From Figure 21, the results show that the data has been fully trained through the Bi-LSTM model. We set total epoch at 1000, each epoch includes 14 batches. In order to save training time, we stopped the training early at the time the precision value of the development set was stopped and once it had increased over 30 epochs. As Figure 21 shows, the models training stops at epoch 150th and epoch 100th. The Dropout layer will reduce the occurrence of overfitting. From the loss function, we can see that overfitting of CoNLL 2003 has been successfully avoided. For the BTC corpus, the model is not completely fully trained. Because the corpus is small, the data cannot support the model to keep improving the performance. After parameter tuning, we tested the uniform and the average to deal with the out of vocabularies (OOV), and the SGD algorithm as the optimization algorithm.

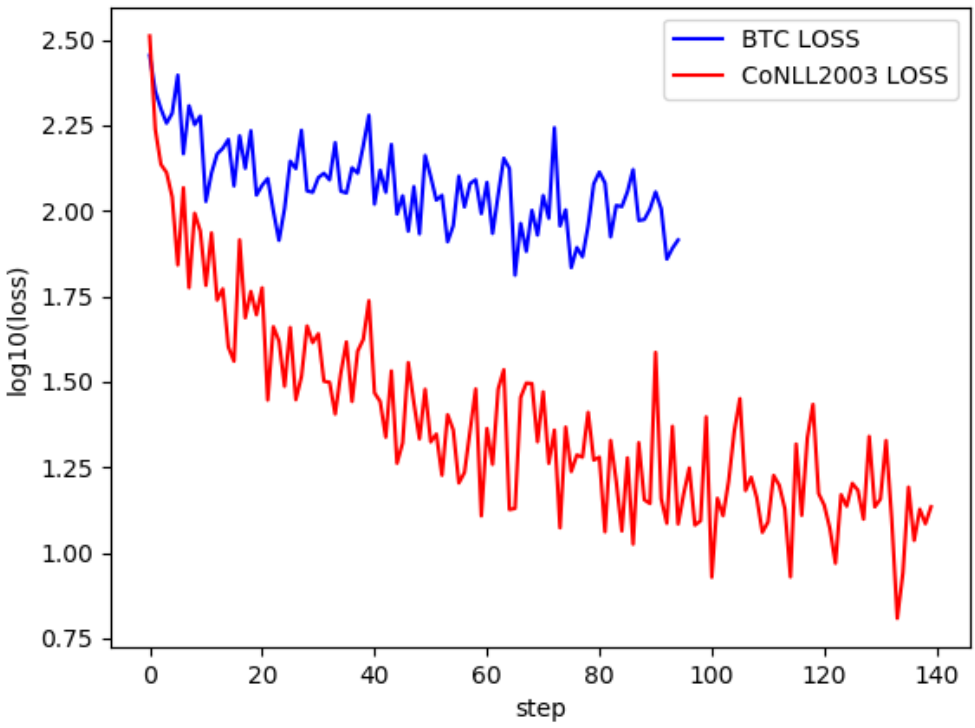

Figure 21: Loss function of corpora

5.2.1 Parameter Tuning

5.2.1.1 Out of Vocabulary (OOV)

GloVe Wikipedia 2014 & Gigaword 5 as pre-trained word vectors are used for training the CoNLL2003 model. GloVe Twitter vectors as pre-trained word vectors are used for the BTC corpus model. For the CoNLL2003 model, we chose 100d vectors, it includes 400000 vocabulary vectors. And 8.85% of the vocabulary in the CoNLL2003 database is Out of Vocabulary (OOV). For the BTC model, we used 100d GloVe Twitter vectors which contains 12K vocabulary vectors. And 37.17% of vocabulary in BTC database is OOV.

There is a huge performance difference between using pre-trained word vectors and without pre-trained word vectors. As table 11 shows, nearly 10% of the F1 difference is related to the use of pre-training vectors or not.

	precision	recall	F1
CoNLL2003 with pre-trained word vectors	90.69%	90.62%	90.66%
CoNLL2003 without pre-trained word vectors	79.43%	79.64%	79.53
BTC with pre-trained word vectors	77.12%	66.31%	71.31%
BTC without pre-trained word vectors	76.61%	51.81%	61.82%

Table 11: Performance of using pre-trained word vectors or not

Regarding OOV, with the exception of using our own embedding method to train the vectors, there are three other methods that we used. We compared these methods and found the best one for our project. We used the BTC database to test the four methods because it contains more OOV than the CoNLL2003 database. The first method is averaging the existing word vectors. The second method is random initialization for each OOV word. The randomly initialized values are between (-0.25, 0.25) or (-0.1, 0.1). The third method is to assign zero for the OOV words.

The results are shown below in Table 12

	precision	recall	F1
average	77.56%	64.23%	70.27%
Initialized (-0.25,	75.31%	65.63%	70.14%

0.25)			
Initialized (-0.1, 0.1)	75.08%	65.69%	70.07%
Zero	78.75%	63.15%	70.09%
Embedding	75.99%	69.55%	72.63%

Table 12: Performance of four OOV methods

5.2.1.2 Parameter Initialization

Initializing the different layers and their weights is very important. Unsuccessful initialization may slow down convergence and affect the final result.
For the CNN layer, we used uniform distribution to initialize the weights of the CNN layer. And then we used the method that is described by Glorot and Bengio (Glorot & Bengio, 2010), the initialization value sample from $\mathrm{N}(0, \mathrm{std})$ where

$$\mathrm{std} = \mathrm{gain} \times \sqrt{\frac{2}{fan_in + fan_out}}$$

Glorot's initialization is used for the CNN layer. The uniform distribution is used for the Bi-LSTM layer and their weights. The initialization of the fully connected layer used the uniform distribution. The weight of the fully connected layer is calculated through the Glorot initialization method.

5.2.1.3 Hyperparameter

The larger filter size will make character-level word embedding better for the longer sentences. Every dataset has its optimal filter size, usually the filter size is set in range one to ten (Zhang & Wallace, 2015). We test 10 values of filter size and found 2 is the best size of the filter for our data. The filter number is the size of the feature of each convolution window. After we tested six values, from 100 to 600, we chose 100.

	precision	recall	F1
1	73.52%	68.86%	71.11%
2	75.99%	69.55%	72.63%
3	74.04%	69.72%	71.81%
4	77.09%	67.94%	72,23%
5	75.35%	62.92%	68.57%
6	75.80%	65.05%	70.02%
7	74.86%	62.51%	68.13%
8	75.79%	62.11%	68.27%

9	75.73%	63.90%	69.31%
10	75.12%	61.89%	68.02%

Table 13: Performance of ten filter sizes

	precision	recall	F1
100	75.99%	69.55%	72.63%
200	75.54%	66.78%	70.89%
300	75.23%	66.02%	70.53%
400	73.49%	62.83%	68.76%
500	72.64%	60.85%	67.63%
600	70.39%	59.56%	65.87%

Table 14: Performance of six filter numbers

Another crucial problem of the neural network is the learning rate (Wei, Xia, Huang, Ni, Dong, Zhao & Yan, 2014). In order to ensure the gradient descent method would perform better, we need to set the value of the learning rate within the appropriate range. If the learning rate is too large, it makes unstable learning for the models, and if it is set at too small a value, the training time will take much too long. The appropriate learning rate can achieve high efficiency under stable training, which can reduce the training time. Learning rate is expressed as the equation.

$$\lambda = \min(\lambda) + [\max(\lambda) - \min(\lambda)] \times e^{\frac{-iteration}{decay-speed}}$$

We use classic for SGD method and Adam. For SGD optimizer, we set the learning rate at 0.001, add momentum as 0.9, the weight decay is 5e-4. For Adam optimizer, we set the learning rate at 0.001.

5.3 Discussing of Four Different Models

This research project used machine learning based methods and the deep neural network based method to deal with the NER task. We can understand the NER task deep through a comparison of these methods.

We know that named the entity recognition task is a sequence labelling problem. Each optimal label of the element depends on the nearby elements. In machine learning, specific algorithms are used to select the globally optimal label sequence for the input sequence. The machine learning methods that we used can be divided into three categories. The HMM model and the CRF model are used to find the probabilities of all of the corresponding categories of the target word and then classifies the word to a class with the highest probability. The HMM model is one of the generative models. It learns joint probability distribution $P(x, y) = P(x|y)P(y)$ through input sequence x and

label sequence y, and then calculates the posterior probability of the samples that belong to each class $P(y|x) = P(x,y)/P(x)$ as the prediction model. It classifies the sample into the class with the highest probability. It combines the two probabilities, we get the equation for prediction:

$$p(y|x) = \frac{P(x|y)P(y)}{p(x)}$$

The HMM model focuses on the generative relationship of the given input x and produces y.

The CRF model is a discriminative model, it models the conditional probability distribution of the label sequence y, and uses it as a prediction model. This model focuses on outputting a label sequence y when the input sequence x is given.

The disadvantages of the conditional random field are slow convergence and long training time. The training speed of the HMM is faster than the conditional random field method. The main reason is that the CRF method requires a lot of time to obtain features based on feature templates.

The SVM model does not build a probability model, it finds the clear decision boundary. The predicted equation of SVM is:

$$\text{sgn}\left(\sum_{i=l}^{l} \alpha_i \, y_i K(x_i^T x)\right) + b$$

The model follows geometric division, dividing the space into multiple parts. It finds and classifies the hyperplanes, and calculates the mapping function of the hyperplanes. Thus, for our NER, it focuses on the single word, through the part-of-speech and chunk features to determine which categories the word belongs to. In the NER task, the context information contains many useful features. The SVM model does not use context information at all. Thus, the SVM model got the lowest performance in the experiment.

The combination model combines multiple models and algorithms, and it uses the output data of the previous layer model as the training data to train the next model. Our Bi-LSTM-CNN-CRF model is a combination model. It uses a CNN model to extract the char level features and combines it with a word embedding and pre-trained vectors as input vectors, then feeds it into two bidirectional LSTM units. Through a fully connected layer, the input vectors are mapped as output vectors such that the dimensions of the output vectors are the number of output labels. Then use the Softmax method normalizes the output as the probability of each label. It then uses the probabilities of the output of BI-LSTM as the state feature vectors add into the CRF layer.

The neural network model can use the powerful nonlinear fitting ability. At training time, it models the complex high- dimensional spatial data through a nonlinear transformation. Then, it uses the model to predict the label sequence of each sentence. However, there is a disadvantage to using the Bi-LSTM model alone. Without the conditional constraint of state transition, the model may output a completely incorrect label sequence. Such as, in

the BIO scheme, in the same chunk entity, the ‘B’ tag must be followed by an ‘I’ tag, it cannot be a ‘B’ tag. The ‘O’ tag must not appear between two ‘I’ tags. Thus, we use the CRF layer to solve this problem. At the same time, using the output of the Bi-LSTM layer as the input of the CRF layer also can avoid the problem of CRF. The biggest problem of the CRF model is the premise of the CRF method. The premise of the CRF method is the current output label is only related to the label of the previous output and the current input. In fact, while we are looking for a named entity in a sentence, it is related to the context. And with the exception of the information of the previous words of the target word, the words behind the target word can also provide a lot of information. Under the above assumption, we lose a lot of context information, which makes the model unable to find the named entity accurately. Also, CRF can learn various features from the given observation sequence, these features are the relationships between the different words. So, the CRF generally learns a rule: B tag followed by I tag, will not appear B tag. This rule will cause the prediction result of CRF so that it will not encounter the error of the above example. Thus, combining the Bi-LSTM and CRF can get the advantages of both models. Due to the project, the most significant difference between machine learning and deep neural network is the feature selection method. Machine learning has high requirements of data, such as, the need for the part of speech or chunk tags. It is necessary to add some handcrafted features to the original data. These features will affect the NLP tasks. According to the differences and characteristics of the data sources, machine learning methods need to consider selecting a feature that can adequately affect the entity. Thus, through pre-processing some features are added to the original data. These features are extracted from the training corpus through the statistical analysis and the linguistic information. The relevant features can be divided into word-level features, char-level features, context features, part-of-speech features, chunking features, gazetteers, prefix and suffix features, semantic features and so on. Without these additional features, machine learning methods get a lower performance. However, compared with the machine learning based method, the neural network based methods are able to get a higher performance without additional features. It can learn the features from the original data through the embedding layer. Thus, the biggest advantage of the neural network based method is to save data pre-processing time, it only needs the original data and the entity label.

5.4 Summary

This Chapter is split into two parts. The first part of the analysis discusses the parameters that may affect the performance of the Bi-LSTM model. The OOV dealing methods, the parameter initialization, and the hyperparameter are introduced in this part. We used precision, recall, and F1 to evaluate the effect of the parameters. The second

part discusses the characteristics of the models. It includes an analysis of the advantages and disadvantages of the four models.

Chapter 6 Conclusion and Future Work

We compared the four models and found the optimal method through the final results of named entity recognition. In addition, we also, answered the research question that we proposed in chapter 1. In this chapter, we will summarize this thesis and introduce the work we will continue doing in the future.

6.1 Conclusion

NER is a necessary task for some downstream NLP tasks. It has a huge effect on these NLP tasks. The purpose of this project is to carry out NER. In fact, it can be translated into a sequence labelling problem. In this project, we trained and tested four different methods and found the best one. These models are based on the statistical methods and deep neural network. After the analysis of results, we summarized the main contribution of this project.

We used the SVM, the HMM, the CRF and the Bi-LSTM-CNN-CRF methods to model and predict the named entities. In order to compare the performance of the models in formal data and noisy, informal data, we select the CoNLL2003 and the BTC corpus to use as our training and testing material. The final results show, the SVM model got 62.12% of F1 for the CoNLL2003 database and 29.82% of F1 for the BTC database. The HMM model got 74.57% of F1 for the CoNLL2003 database and 29.82% of F1 for the BTC database. The CRF model got 80.94% of F1 for CoNLL2003 database and 29.82% of F1 for the BTC database. The Bi-LSTM model got 89.97% of F1 for the CoNLL2003 database and 71.31% of F1 for the BTC database. Through parameter tuning, the Bi-LSTM model got the best results in both corpora. Also, for the informal content corpus, the performance of the four models are lower than their performance in the formal content corpus.

This project through the experiment built four models that compared the deep learning model and the machine learning models in NER. In our experiment, the deep learning model is better than the statistical-based model. We also analysed the models, and discussed the advantages and disadvantages of the models.

6.2 Future Works

Our future work will involve the following:

- Use focal loss to deal with extremely unbalanced data, such as, BTC corpus
- Research the relationship between the number of embedding dimensions and the number of hidden layers in the LSTM layer
- Research the impact of the different initialization distribution on the Bi-LSTM
- Research the impact of the different window sizes on the CNN layer
- Research the most suitable unsupervised learning method for NER

Reference

Aroonmanakun, W., Nupairoj, N., Muangsin, V., & Choemprayong, S. (2018). Thai Monitor Corpus: Challenges and Contribution to Thai NLP. *Vacana*, *6*(2), 1-14.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baldwin, T., de Marneffe, M. C., Han, B., Kim, Y. B., Ritter, A., & Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text* (pp. 126-135).

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137-1155.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157-166.

Bharadwaj, A., Mortensen, D., Dyer, C., & Carbonell, J. (2016). Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 1462-1472).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.

Bonadiman, D., Severyn, A., & Moschitti, A. (2015). Deep neural networks for named entity recognition in Italian. *CLiC it*, *51*.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 27.

Chang, J. T., Schütze, H., & Altman, R. B. (2004). GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics*, *20*(2), 216-225.

Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357-370.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Choi, Y., & Cha, J. (2016). Korean Named Entity Recognition and Classification using Word Embedding Features. *Journal of KIISE*, *43*(6), 678-685.

Chrupala, G. (2011). Efficient induction of probabilistic word classes with LDA. In *Proceedings of 5th International Joint Conference on Natural Language Processing* (pp. 363-372).

Clark, K., Luong, M. T., Manning, C. D., & Le, Q. V. (2018). Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.

Collobert, R. (2011). Deep learning for efficient discriminative parsing. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*(pp. 224-232).

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(Aug), 2493-2537.

Cotik, V., Rodríguez, H., & Vivaldi, J. (2018). Spanish named entity recognition in the biomedical domain. In *Annual International Symposium on Information Management and Big Data* (pp. 233-248). Springer, Cham.

Daelemans, W., & Van den Bosch, A. (2005). *Memory-based language processing*. Cambridge University Press.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., ... & Ng, A. Y. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems* (pp. 1223-1231).

Derczynski, L., Maynard, D., Rizzo, G., Van Erp, M., Gorrell, G., Troncy, R., ... & Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, *51*(2), 32-49.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, N., & Nguyen, K. A. (2018). Attentive Neural Network for Named Entity Recognition in Vietnamese. *arXiv preprint arXiv:1810.13097*.

Dozat, T. (2016). Incorporating nesterov momentum into adam.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121-2159.

Dugas, F., & Nichols, E. (2016). DeepNNNER: Applying BLSTM-CNNs and Extended Lexicons

to Named Entity Recognition in Tweets. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)* (pp. 178-187).

Ebersbach, M., Herms, R., Lohr, C., & Eibl, M. (2016). Wrappers for Feature Subset Selection in CRF-based Clinical Information Extraction. In *CLEF (Working Notes)* (pp. 69-80).

Ekbal, A., & Bandyopadhyay, S. (2008). Bengali named entity recognition using support vector machine. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115.

Finkel, J., Dingare, S., Manning, C. D., Nissim, M., Alex, B., & Grover, C. (2005). Exploring the boundaries: gene and protein identification in biomedical text. *BMC bioinformatics*, *6*(1), S5
Fleischman, M. (2001). Automated subcategorization of named entities. In *ACL (Companion Volume)* (pp. 25-30).

Fleischman, M., & Hovy, E. (2002). Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.

Florian, R., Ittycheriah, A., Jing, H., & Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (pp. 168-171). Association for Computational Linguistics.

Galibert, O., Rosset, S., Grouin, C., Zweigenbaum, P., & Quintard, L. (2012). Extended named entity annotation on ocred documents: From corpus constitution to evaluation campaign. In *Proceedings of the Eighth conference on International Language Resources and Evaluation (LREC'12), Istanbul, Turkey, may*.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* (Vol. 3, pp. 189-194). IEEE.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov models: applications in computer vision* (pp. 9-41).

Ghosh, S., Ghosh, S., & Das, D. (2016). Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching* (pp. 90-97).

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256).

Godin, F., Vandersmissen, B., De Neve, W., & Van de Walle, R. (2015). Multimedia Lab $@ $ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. In *Proceedings of the Workshop on Noisy User-generated Text* (pp. 146-153).

Goller, C., & Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)* (Vol. 1, pp. 347-352). IEEE.

Gómez-Chova, L., Camps-Valls, G., Munoz-Mari, J., & Calpe, J. (2008). Semisupervised image classification with Laplacian support vector machines. *IEEE Geoscience and Remote Sensing Letters*, *5*(3), 336-340.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5-6), 602-610.

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). IEEE.

Guo, J., Xu, G., Cheng, X., & Li, H. (2009). Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (pp. 267-274). ACM.

Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., & Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, *33*(14), i37-i48.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level

performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hemati, W., & Mehler, A. (2019). LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools. *Journal of cheminformatics*, *11*(1), 3.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., ... & Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, *29*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, *13*(2), 415-425.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Isozaki, H. (2001). Japanese named entity recognition based on a simple rule generator and decision tree learning. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (pp. 314-321). Association for Computational Linguistics.

Isozaki, H., & Kazawa, H. (2002). Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.

Jagannatha, A. N., & Yu, H. (2016). Bidirectional RNN for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting* (Vol. 2016, p. 473). NIH Public Access.

Jarrett, K., Kavukcuoglu, K., & LeCun, Y. (2009). What is the best multi-stage architecture for

object recognition?. In *2009 IEEE 12th international conference on computer vision* (pp. 2146-2153). IEEE.

Jensen, L. J., Saric, J., & Bork, P. (2006). Literature mining for the biologist: from information retrieval to biological discovery. *Nature reviews genetics*, *7*(2), 119.

Ji, Y., Tong, C., Liang, J., Yang, X., Zhao, Z., & Wang, X. (2019). A deep learning method for named entity recognition in bidding document. In *Journal of Physics: Conference Series* (Vol. 1168, No. 3, p. 032076). IOP Publishing.

Johansen, A. R., Sønderby, C. K., Sønderby, S. K., & Winther, O. (2017). Deep recurrent conditional random field network for protein secondary prediction. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 73-78). ACM.

Ju, Z., Wang, J., & Zhu, F. (2011). Named entity recognition from biomedical text using SVM. In *2011 5th international conference on bioinformatics and biomedical engineering* (pp. 1-4). IEEE.

Junaida, M. K., Jayan, J. P., & Sherly, E. (2017). Word Sense Disambiguation for Malayalam in a Conditional Random Field Framework. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)* (pp. 495-502).

Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3128-3137).

Kazama, J. I., Makino, T., Ohta, Y., & Tsujii, J. I. (2002). Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3* (pp. 1-8). Association for Computational Linguistics.

Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation*, *15*(7), 1667-1689.

Kim, J. D., Ohta, T., Tsuruoka, Y., Tateisi, Y., & Collier, N. (2004). Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications* (pp. 70-75). Association for Computational Linguistics.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems* (pp. 3581-3589).

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Klein, D., Smarr, J., Nguyen, H., & Manning, C. D. (2003). Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (pp. 180-183). Association for Computational Linguistics.

Konkol, M., Brychcín, T., & Konopík, M. (2015). Latent semantics in named entity recognition. *Expert Systems with Applications*, *42*(7), 3470-3479.

Krebel, U. G. (1999). Pairwise classification and support vector machines. *Advances in kernel methods: support vector learning*, 255-268.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Leaman, R., & Gonzalez, G. (2008). BANNER: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008* (pp. 652-663).

Lebret, R., Grangier, D., & Auli, M. (2016). Generating text from structured data with application to the biography domain. *ArXiv e-prints, March*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lee, C., Hwang, Y. G., Oh, H. J., Lim, S., Heo, J., Lee, C. H., ... & Jang, M. G. (2006). Fine-grained named entity recognition using conditional random fields for question answering. In *Asia Information Retrieval Symposium* (pp. 581-587). Springer, Berlin, Heidelberg.

Lee, H., Eum, S., & Kwon, H. (2017). ME R-CNN: Multi-Expert R-CNN for Object Detection. *arXiv preprint arXiv:1704.01069*.

Lee, K. J., Hwang, Y. S., Kim, S., & Rim, H. C. (2004). Biomedical named entity recognition using two-phase model based on SVMs. *Journal of Biomedical Informatics*, *37*(6), 436-447.

Lee, S., & Lee, G. G. (2005). Heuristic methods for reducing errors of geographic named entities learned by bootstrapping. In *International Conference on Natural Language Processing* (pp. 658-

669). Springer, Berlin, Heidelberg.

Leslie, C., Eskin, E., & Noble, W. S. (2001). The spectrum kernel: A string kernel for SVM protein classification. In *Biocomputing 2002* (pp. 564-575).

Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., & Lee, B. S. (2012). Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 721-730). ACM.

Li, E., Femiani, J., Xu, S., Zhang, X., & Wonka, P. (2015). Robust rooftop extraction from visible band images using higher order CRF. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(8), 4483-4495.

Li, L., Jin, L., Jiang, Z., Song, D., & Huang, D. (2015). Biomedical named entity recognition based on extended recurrent neural networks. In *2015 IEEE International Conference on bioinformatics and biomedicine (BIBM)* (pp. 649-652). IEEE.

Limsopatham, N., & Collier, N. H. (2016). Bidirectional LSTM for named entity recognition in Twitter messages.

Lin, B. Y., Xu, F., Luo, Z., & Zhu, K. (2017). Multi-channel bilstm-crf model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text* (pp. 160-165).

Lin, H. T., & Lin, C. J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *submitted to Neural Computation*, *3*, 1-32.

Ling, X., & Weld, D. S. (2012). Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Liu, F., Lin, G., & Shen, C. (2015). CRF learning with CNN features for image segmentation. *Pattern Recognition*, *48*(10), 2983-2992.

Liu, J., & Birnbaum, L. (2007). Measuring semantic similarity between named entities by searching the web directory. In *Proceedings of the IEEE/WIC/ACM international Conference on Web intelligence* (pp. 461-465). IEEE Computer Society.

Liu, S., Yang, N., Li, M., & Zhou, M. (2014). A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 1491-1500).

Loyola, P., Marrese-Taylor, E., & Matsuo, Y. (2017). A neural architecture for generating natural language descriptions from source code changes. *arXiv preprint arXiv:1704.04856*.

Luu, T. M., Phan, R., Davey, R., & Chetty, G. (2018). Clinical Name Entity Recognition Based on

Recurrent Neural Networks. In *2018 18th International Conference on Computational Science and Applications (ICCSA)* (pp. 1-9). IEEE.

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mai, K., Pham, T. H., Nguyen, M. T., Duc, N. T., Bollegala, D., Sasano, R., & Sekine, S. (2018). An empirical study on fine-grained named entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 711-722).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Minar, M. R., & Naher, J. (2018). Recent Advances in Deep Learning: An Overview. *arXiv preprint arXiv:1807.08169*.

Mnih, A., & Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning* (pp. 641-648). ACM.

Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems* (pp. 1081-1088).

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, *13*(6), 47-60.

Morales-Cordovilla, J. A., Sanchez, V., & Ratajczak, M. (2018). Protein alignment based on higher order conditional random fields for template-based modeling. *PloS one*, *13*(6), e0197912.

Morwal, S., Jahan, N., & Chopra, D. (2012). Named entity recognition using hidden Markov model (HMM). *International Journal on Natural Language Computing (IJNLC)*, *1*(4), 15-23.

Mozharova, V., & Loukachevitch, N. (2016). Two-stage approach in Russian named entity recognition. In *2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)* (pp. 1-6). IEEE.

Nadeau, D. (2007). Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision.

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544-551

Narayanaswamy, M., Ravikumar, K. E., & Vijay-Shanker, K. (2002). A biological named entity recognizer. In *Biocomputing 2003* (pp. 427-438).

Özgür, A., Özgür, L., & Güngör, T. (2005). Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences* (pp. 606-615). Springer, Berlin, Heidelberg.

Palmer, D. D., & Day, D. S. (1997). A statistical profile of the named entity task. In *Fifth Conference on Applied Natural Language Processing*.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310-1318).

Petasis, G., Vichot, F., Wolinski, F., Paliouras, G., Karkaletsis, V., & Spyropoulos, C. D. (2001). Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (pp. 426-433). Association for Computational Linguistics.

Peters, M. E., Ammar, W., Bhagavatula, C., & Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pham, T. H., & Le-Hong, P. (2017). End-to-end recurrent neural network models for vietnamese named entity recognition: Word-level vs. character-level. In *International Conference of the Pacific Association for Computational Linguistics* (pp. 219-232). Springer, Singapore.

Ponomareva, N., Pla, F., Molina, A., & Rosso, P. (2007). Biomedical named entity recognition: a poor knowledge HMM-based approach. In *International Conference on Application of Natural Language to Information Systems* (pp. 382-387). Springer, Berlin, Heidelberg.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257-286.

Rabiner, L. R., & Juang, B. H. (1986). An introduction to hidden Markov models. *ieee assp magazine*, *3*(1), 4-16.

Ramachandran, P., Liu, P. J., & Le, Q. V. (2016). Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Ramage, D., Hall, D., Nallapati, R., & Manning, C. D. (2009). Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1* (pp. 248-256). Association for Computational Linguistics.

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., & Pande, V. (2015). Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*.

Ratinov, L., & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning* (pp. 147-155). Association for Computational Linguistics.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Rei, M., Crichton, G. K., & Pyysalo, S. (2016). Attending to characters in neural sequence labeling models. *arXiv preprint arXiv:1611.04361*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Ritter, A., Clark, S., & Etzioni, O. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 1524-1534). Association for Computational Linguistics.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Salleh, M. S., Asmai, S. A., Basiron, H., & Ahmad, S. (2017). A Malay Named Entity Recognition using conditional random fields. In *2017 5th International Conference on Information and Communication Technology (ICoIC7)* (pp. 1-6). IEEE.

Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Santos, C. D., & Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 1818-1826).

Santos, C. D., & Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp.

1818-1826).

Santos, C. N. D., & Guimaraes, V. (2015). Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85-117.

Şeker, G. A., & Eryiğit, G. (2017). Extending a CRF-based named entity recognition model for Turkish well formed text and user generated content 1. *Semantic Web*, *8*(5), 625-642.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Shen, D., Zhang, J., Su, J., Zhou, G., & Tan, C. L. (2004). Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (p. 589). Association for Computational Linguistics.

Si, L., Kanungo, T., & Huang, X. (2005). Boosting performance of bio-entity recognition by combining results from multiple systems. In *Proceedings of the 5th international workshop on Bioinformatics* (pp. 76-83). ACM.

Smith, L., Tanabe, L. K., nee Ando, R. J., Kuo, C. J., Chung, I. F., Hsu, C. N., ... & Torii, M. (2008). Overview of BioCreative II gene mention recognition. *Genome biology*, *9*(2), S2.

Smits, G. F., & Jordaan, E. M. (2002). Improved SVM regression using mixtures of kernels. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)* (Vol. 3, pp. 2785-2790). IEEE.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, *14*(3), 199-222.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427-437.

Srihari, R. (2000). A hybrid approach for named entity and sub-type tagging. In *Sixth Applied Natural Language Processing Conference*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 1017-

1024).

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

Sutton, C., & McCallum, A. (2012). An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, *4*(4), 267-373.

Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, *9*(3), 293-300.

Tang, B., Cao, H., Wang, X., Chen, Q., & Xu, H. (2014). Evaluating word representation features in biomedical named entity recognition tasks. *BioMed research international*, *2014*.

Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1422-1432).

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence* (pp. 485-492). Morgan Kaufmann Publishers Inc..

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384-394). Association for Computational Linguistics.

Varma, K. I., Krishnamoorthy, S., & Pisipati, R. K. (2016). *U.S. Patent No. 9,280,535*. Washington, DC: U.S. Patent and Trademark Office.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*(pp. 5998-6008).

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).

Wallach, H. M. (2004). Conditional random fields: An introduction. *Technical Reports (CIS)*, 22.

Wang, Y., Bai, H., Stanton, M., Chen, W. Y., & Chang, E. Y. (2009). Plda: Parallel latent dirichlet allocation for large-scale applications. In *International Conference on Algorithmic Applications in Management* (pp. 301-314). Springer, Berlin, Heidelberg.

Wei, Y., Xia, W., Huang, J., Ni, B., Dong, J., Zhao, Y., & Yan, S. (2014). Cnn: Single-label to multi-label. *arXiv preprint arXiv:1406.5726*.

Wibawa, A. S., & Purwarianti, A. (2016). Indonesian named-entity recognition for 15 classes using ensemble supervised learning. *Procedia Computer Science*, *81*, 221-228.

Wu, Y., Xu, J., Jiang, M., Zhang, Y., & Xu, H. (2015). A study of neural word embeddings for named entity recognition in clinical text. In *AMIA Annual Symposium Proceedings* (Vol. 2015, p. 1326). American Medical Informatics Association.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

Yamada, H., & Matsumoto, Y. (2001). Applying support vector machine to multi-class classification problems. *IPSJ SIG Notes NL-146*, *6*.

Yeniterzi, R., Tür, G., & Oflazer, K. (2018). Turkish Named-Entity Recognition. In *Turkish Natural Language Processing*(pp. 115-132). Springer, Cham.

Yogatama, D., Gillick, D., & Lazic, N. (2015). Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Vol. 2, pp. 291-296).

Zagal, J. P., Tomuro, N., & Shepitsen, A. (2012). Natural language processing in game studies research: An overview. *Simulation & Gaming*, *43*(3), 356-373.

Zhang, H. P., Yu, H. K., Xiong, D. Y., & Liu, Q. (2003). HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17* (pp. 184-187). Association for Computational Linguistics.

Zhang, J., Shen, D., Zhou, G., Su, J., & Tan, C. L. (2004). Enhancing HMM-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics*, *37*(6), 411-422.

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Zhang, Y., Brady, M., & Smith, S. (2001). Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, *20*(1), 45-57.

Zhou, G., & Su, J. (2002). Named entity recognition using an HMM-based chunk tagger.

In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*(pp. 473-480). Association for Computational Linguistics.

Zhu, F., & Shen, B. (2012). Combined SVM-CRFs for biological named entity recognition with maximal bidirectional squeezing. *PloS one*, *7*(6), e39230.

Zhu, X. J. (2005). *Semi-supervised learning literature survey*. University of Wisconsin-Madison Department of Computer Sciences.

Zirikly, A., & Diab, M. (2015). Named entity recognition for arabic social media. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing* (pp. 176-185).