DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Explainable Deep Learning for Natural Language Processing

JIN HU

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Explainable Deep Learning for Natural Language Processing

JIN HU

Master in ICT Innovation
Date: October 23, 2018
Supervisor: Anne Håkansson
Examiner: Mihhail Matskin
Swedish title: förklaras djupt lärande för naturlig språkbehandling
School of Information and Communication Technology

Abstract

Deep learning methods get impressive performance in many Natural Neural Processing (NLP) tasks, but it is still difficult to know what happened inside a deep neural network. In this thesis, a general overview of Explainable AI and how explainable deep learning methods applied for NLP tasks is given. Then the Bi-directional LSTM and CRF (Bi-LSTM-CRF) model for Named Entity Recognition (NER) task is introduced, as well as the approach to make this model explainable. The approach to visualize the importance of neurons in Bi-LSTM layer of the model for NER by Layer-wise Relevance Propagation (LRP) is proposed, which can measure how neurons contribute to each prediction of a word in a sequence. Ideas about how to measure the influence of CRF layer of the Bi-LSTM-CRF model is also described.

Sammanfattning

Djupa inlärningsmetoder får imponerande prestanda i många naturliga Neural Processing (NLP) uppgifter, men det är fortfarande svårt att veta vad hände inne i ett djupt neuralt nätverk. I denna avhandling, en allmän översikt av förklarliga AI och hur förklarliga djupa inlärningsmetoder tillämpas för NLP-uppgifter ges. Då den bi-riktiga LSTM och CRF (BiLSTM-CRF) modell för Named Entity Recognition (NER) uppgift införs, liksom tillvägagångssättet för att göra denna modell förklarlig. De tillvägagångssätt för att visualisera vikten av neuroner i Bi-LSTM-skiktet av Modellen för NER genom Layer-Wise Relevance Propagation (LRP) föreslås, som kan mäta hur neuroner bidrar till varje förutsägelse av ett ord i en sekvens. Idéer om hur man mäter påverkan av CRF-skiktet i Bi-LSTM-CRF-modellen beskrivs också.

Contents

List of Tables

List of Figures

Abbreviations

AI Artificial Intelligence.

Bi-LSTM Bi-directional LSTM.

Bi-LSTM-CRF Bi-directional LSTM and CRF.

CBOW Continuous Bag-of-Words Model.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

LIME Local Interpretable Model-Agnostic Explanations.

LRP Layer-wise Relevance Propagation.

LSTM Long Short-Term Memory.

MSE Mean Square Error.

NER Named Entity Recognition.

NLP Natural Language Processing.

NN Neural Network.

RNN Recurrent Neural Network.

t-SNE t-Distributed Stochastic Neighbor Embedding.

XAI Explainable Artificial Intelligence.

Chapter 1

Introduction

In this project, we first investigate several methods to explain the behavior of deep neural networks, and their applications in Natural Language Processing (NLP) field. We also implement the explainable methods with the combination of Bi-directional LSTM and CRF (Bi-LSTM-CRF) used in Named Entity Recognition (NER) as well. This project was carried out at Seavus Stockholm AB, Sweden.

1.1 Motivation

Machine learning methods have been applied widely in many fields like recommender system [1], chatbots [2], and self-driving cars [3]. However, some machine learning methods are opaque, non-intuitive, and difficult for people to understand, the effectiveness of machine learning are limited.

In recent years, deep learning methods, as a type of machine learning methods, had yielded state-of-art performance in many problems. But it is possible that people worry about how to make sure a deep learning model makes the right decision when huge numbers of parameters, many rounds of iteration and optimization from deep learning models are taken into consideration. For example, many medical systems apply deep learning techniques to help diagnosis, as Wang et al. [4] uses it to detect breast cancer, but the result needs to be checked by human doctors in order to make sure the diagnosis is reasonable. The more complicated the model is, the more difficult to explain how the result comes out so that people probably suspect the prediction. If the model can explain itself, it will gain more trust from users, and be more convenient to debug or improve for developers. Because of needs for trust and improvements, many researchers devote themselves to developing explainable machine learning methods.

Explainable machine learning attracts attention of organizations such as Defense Advanced Research Projects Agency (DARPA), which launched the Explainable Artificial Intelligence (XAI) [5], as well as companies focus on Artificial Intelligence (AI) like Google, which initiated People +AI Research (PAIR)[1]. In academia, researchers conducted a lot of methods to make deep learning models more transparent, which include works on revealing the inner structure and behavior of the model, as well as generating explainable interfaces which can be accepted by normal users.

Considering more and more applications will be combined with AI, and this issue has been attached importance gradually by both academia and industry, the explainability will become an important characteristic of a machine learning model in the future.

1.2 Problem Statement

There are many important applications in Natural Language Processing field widely used by people, like machine translation, speech recognition, and information retrieval. With the development of deep learning, more and more NLP tasks reached impressive performance by applying deep learning methods. Correspondingly, different explainable AI methods are produced to make how the deep learning model works more transparent and understandable in NLP tasks like sentiment analysis. Named Entity Recognition is a basic task in NLP field, it is also used in many applications like search engine and online advertisement. As a basic and important task in NLP field, though there are many works focus on how to apply deep neural networks to solve NER problem, but not much work about how to apply explainable deep learning methods to NER. To limit the scope of the thesis and meet the needs of Seavus Stockholm AB, the following problem should be considered in this thesis:

How to make the Bi-LSTM-CRF model used in Named Entity Recognition explainable for people who have machine learning background?

[1] https://ai.google/research/teams/brain/pair

1.3 Purpose

After some literature review, the purpose of this thesis project is stated as: Investigate different explainable deep learning methods for NLP, implement an appropriate explainable method for the named entity recognizer based on Bi-LSTM-CRF, and make people who have some background knowledge get some explainability from the visualization.

1.4 Goals

The goal of this thesis is to help developers at Seavus Stockholm AB to debug and evaluate the performance of their product in the NER field, and provide comparisons of different explainable methods so that they can choose when they develop AI solutions in the future.

1.5 Sustainability and Ethics

Research on explainable machine learning methods could be beneficial for AI products to gain trust from users, as well as for developers to get better results from the model. Considering privacy issues, we use a public dataset to do experiments, more details about the dataset are described in Section 4.1.

From the sustainability aspect, there is no direct impact on the environment in developing the explainable AI system. According to our knowledge, developing this system will not cause obvious pollution or waste. Though it consumes computer hardware resources and electricity, the energy consumption is negligible.

From the ethics aspect, research on explainable machine learning model could help to alleviate the anxiety due to the confusion caused by the black-box model when the model gets into use in people's daily life. Developing explainable machine learning system is beneficial to make the model get correct and appropriate results, especially in important fields people who do not have much background knowledge about machine learning, like medical and crime justification. Explainable AI systems are also helpful for increasing the adoption of machine learning because models with explanation can obtain more trust from users.

1.6 Research Methodology

The research methodology employed by this thesis is described in [6]. The literature study was conducted to find ideas to solve the problem from recent related works. The implementation of the algorithm was done in an exploratory and iterative way. After the experiments, the inductive approach will be used to analyze the results and find the possible patterns of how the model thinks. We also discuss the pros and cons of different approaches, to make recommendations about when to use what model and provide suggestions for future work.

The programming language is Python. The model was implemented by Keras, more details about the tool and libraries used in this thesis project is described in Section 4.2.

1.7 Project Environment

This thesis project was conducted in collaboration with Seavus Stockholm AB. Seavus is a software development and consulting company with a proven track record in providing successful enterprise-wide business solutions. I joined a project about NER in medical field, and was responsible for providing some explainability for the named entity recognizer.

1.8 Delimitations

The research of explainable AI includes two aspects, namely explainable model and explainable interface. It is important to design methods to make the explanation acceptable for normal users, however, this is out of scope of this thesis. This thesis focuses on developing explainable model in NLP field for people who have technology background.

1.9 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 shows the background theories used in this thesis and discusses the related works. Chapter 3 gives a more specific description about how to make explainable NER, which includes details of the model. Chapter 4 represents the introduction of the public dataset and configurations of the experiment environment. The result from the explainable NER is demonstrated to see how to visualize the inner states of the model and how input variables contribute to the generated result. Chapter 5 draws the conclusion of the thesis.

Chapter 2

Background and Related Works

2.1 Machine Learning and Deep Learning

Machine learning, as a subset of AI, attracts the attention of many researchers. It often ’learns’ some implicit rules from labeled data and makes predictions based on some statistical techniques [7]. Machine learning has been applied in many research fields such as Computer Vision (CV) [8] and Natural Language Processing (NLP) [9]. As a thriving research field, there are also a lot of practical applications applying machine learning techniques, such as machine translation [10], face recognition [11], and so forth. Machine learning includes tasks like supervised learning and unsupervised learning. In supervised learning, the model can learn possible patterns from labeled data, while in unsupervised learning, the model could just find structure or rules from input unlabeled data on its own.

Due to more accessible hardware and the improvements of GPUs, deep learning has been explored by researchers in recent years. Deep learning methods try to imitate the way in which human brain works, it consists of hidden layers consist of multiple neurons. Using deep learning can get good performance in several applications such as playing go [12] and speech recognition [13] without manual feature engineering and domain expertise.

2.1.1 Neural Networks

A Neural Network (NN) is a machine learning model which imitates the function and structure of the human brain. It includes simple computational nodes called *neurons* or *units*. How a single neuron works are demonstrated in Figure 2.1. This neuron receives information along with the incoming edges which are sent from x_1, x_2 and x_3, and computes how much it receives by the weight of the edge, by applying a non-linear activation function to the sum of all it receives, it produces an output as its activation value. Suppose $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is the weight vector, b is the bias, f is the activation function, $\odot$ means element-wise multiplication, the output z can be calculated by equation 2.1.

$$z = f(\boldsymbol{w} \odot \boldsymbol{x} + b) \tag{2.1}$$

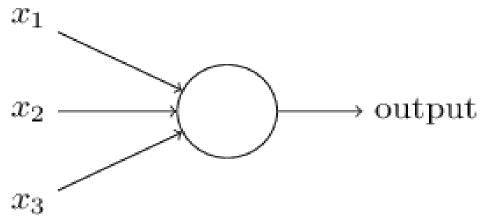


Figure 2.1: Structure of a single neuron [14]

There are several activation functions often used, such as *Sigmoid* (σ), *Rectified Linear Units (ReLU)*, and *tanh*. These functions are defined by equation 2.2, 2.3 and 2.4.

$$\sigma(z) = \frac{1}{1 + e^{(-z)}} \tag{2.2}$$

$$ReLU(z) = max(0, z) \tag{2.3}$$

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.4}$$

Figure 2.2 shows an example of a multi-layer NN. Multiple neurons form layers of units, each neuron is connected to all neurons in the previous layer by edges with different weights, and several connected layers compose the neural network. The first layer is the *input layer* which receive inputs, and the last layer which yields the output is called the *output layer*, the intermediate layers are regarded as *hidden layers*. When information comes in, it flows from the input layer to the output layer and lights the correspond neuron in the output layer to indicate the result.

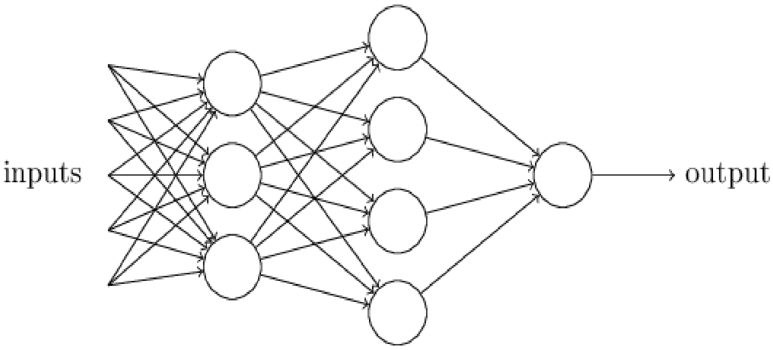


Figure 2.2: structure of a multi-layer NN [14]

To measure the difference between the output made by the neural network and the expected output, the *Mean Square Error (MSE)* loss function is often used. It evaluates the average squared distance between predictions and true values. The equation of MSE is indicated as 2.5, N represents the number of samples used by the model, y denotes the true value while $\hat{y}$ means the value yielded by the neural network.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.5}$$

Then the learning problem is converted to an optimization problem which aims to minimize the loss function by adjusting parameters in the neural network. The optimization method used is called *Gradient Descent*. The gradient measures the changes of loss value corresponding to the small changes of parameters, it is calculated as the partial derivatives of parameters regarding the loss function. Suppose L is the loss function, θ are parameters, η is a small positive number called *learning rate* to control the pace to tune the parameters, we can update parameters by equation 2.6 iteratively, so that we keep decreasing the value of loss function until we reach a global minimum.

$$\theta = \theta - \eta\frac{\partial L(\theta)}{\partial \theta} \tag{2.6}$$

Usually, there are a lot of training samples, so it is not an efficient way to calculate gradient separately for each training input. To deal with this issue, we choose a small number of training samples randomly, each set of training samples is referred as a *mini-batch*, and then we update parameters after calculating the loss of each mini-batch.

This method is called *Stochastic Gradient Descent (SGD)*. More details about training methods can be found in [15].

2.1.2 Recurrent Neural Networks

Standard RNNs

Traditional neural networks do not consider the relations and the order of the input data. For example, if a NN is trying to classify what happened in a video, since the traditional NN deal with sample data individually, it cannot use the former frames to help understand later ones. Recurrent Neural Network (RNN) was proposed to solve sequence problems by taking the continuity of individual data sample into consideration. In units of RNN, there is a loop to keep the history information, and convey the information from previous hidden state to the next. Figure 2.3 shows an example of RNN. There is a single unit of RNN on the left side of this figure, the loop of this unit can convey the state at current timestep to the next timestep. In each timestep, the input x_t RNN generates a hidden state h_t. According to the right side of figure 2.3, the loop operation of a single unit in RNN can be regarded as a sequence of copies of the same unit in chronological order, each copy passes information to the copy in next timestep. There are several variations of RNN depends on different needs. In this thesis, we will use Long Short-Term Memory (LSTM).

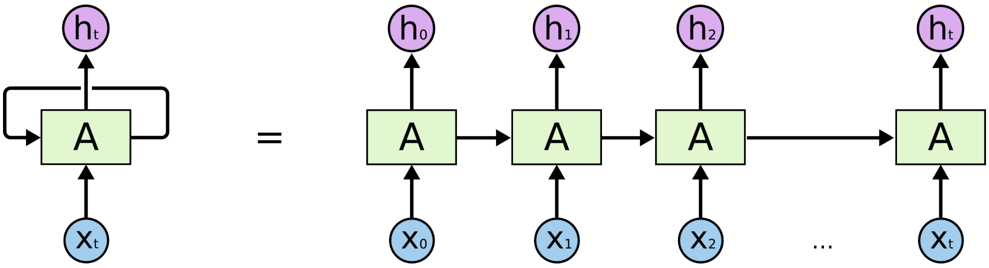


Figure 2.3: Structure of RNN [16]

LSTM

RNNs can keep the memory of every timestep before the current one to help predict the result. For example, to predict the next word of a text, we only need to consider a few adjacent words, the word to be predicted has little distance from its related words. In this case, the RNN works. But what if the relevant information is far away from the word to be predicted? It is difficult for standard RNNs to associate the information to be predicted and its relevant information once the gap between them is increasing.

To solve the problem caused by "long-term dependencies", Hochreiter and Schmidhuber [17] proposed Long Short-Term Memory which aims to remember information for short term. Comparing to standard RNNs, LSTM performs better at overcoming vanishing gradients problem by using the unit with more complex architecture as shown in Figure 2.4. There are three gates designed to control how the information pass through the cell. These three gates are described as follows:

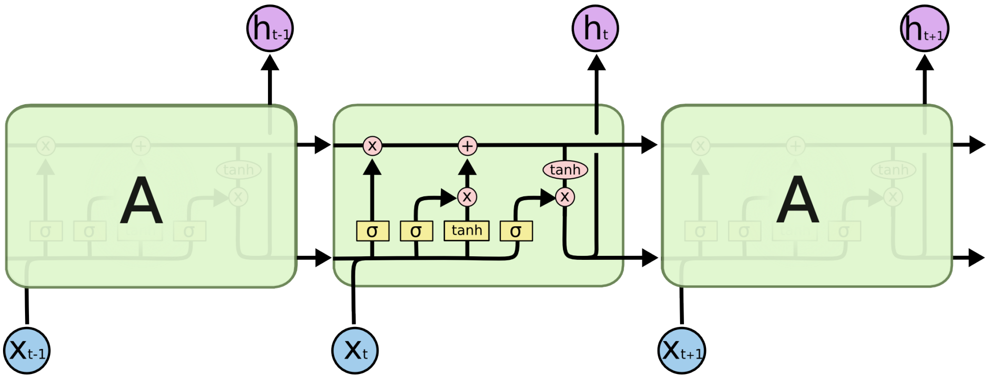


Figure 2.4: Structure of a unit in LSTM [16]

- Forget gate: The forget gate controls how the information from the current input x_t and the output from the previous time step h_{t-1} flow in the current cell, after calculated by an activation function σ. Following equation 2.7 it outputs a vector f_t with all elements between 0 and 1. This vector points which information is allowed to pass.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.7}$$

- Input gate: The input gate decides how much new input information should be added to the cell state. First, a $sigmoid$ function decides which information needs to be updated, and a $tanh$ function generalizes the $\hat{C}_t$ which means the contents available for update. Then, the old cell state C_{t-1} can be replaced by adding new information into the cell state and get C_t. This process is represented by equations 2.8:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \hat{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \hat{C}_t \end{aligned} \tag{2.8}$$

- Output gate: What to output depends on the output gate. The output gate will filter the cell state C_t and output the hidden state h_t at current timestep. As equations 2.9 states, it uses a $sigmoid$ layer to calculate a vector decides which part of the information in C_t is allowed to output (o_t). o_t is multiplied by C_t to get the final result.

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{2.9}$$

***Note:** in equations 2.7, 2.8, and 2.9, W_* and b_* means weights and biases for the same cell in different time step respectively.*

Bi-directional LSTM

Standard LSTM only considers the influence of past information. Sometimes future feature can also influence the prediction. Bi-directional LSTM (Bi-LSTM) [18] enables the use of both past features and future features for a certain time step. The structure of Bi-LSTM is showed in Figure 2.5. For example, if a input sample sentence has 10 words (10 timesteps) $x_1, x_2, \cdots, x_{10}$, there exists two separate LSTM cells, the Bi-LSTM network works as follows:

1. For forward-direction LSTM cells, the order of input words is $x_1, x_2, \cdots, x_{10}$, the output is a set of hidden states $\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_{10}}$;

2. For backward-direction LSTM cells, the order of input words is $x_{10}, x_9, \cdots, x_1$, and output the hidden states set $\overleftarrow{h_{10}}, \overleftarrow{h_9}, \cdots, \overleftarrow{h_1}$;

3. Concatenate two sets of hidden state as $[\overrightarrow{h_1}, \overleftarrow{h_1}], [\overrightarrow{h_2}, \overleftarrow{h_2}], \cdots, [\overrightarrow{h_{10}}, \overleftarrow{h_{10}}]$

4. For input x_t at each time step, get output states $H_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$, then the rest operations are as same as in LSTM network.

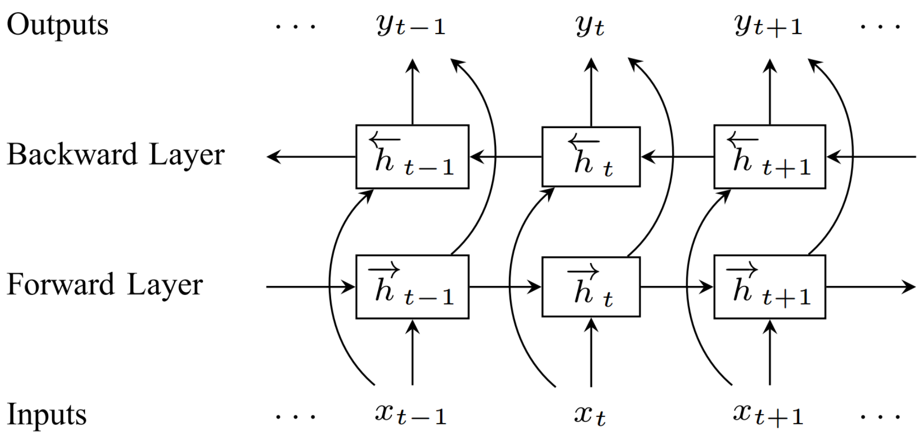

Outputs
Backward Layer
Forward Layer
Inputs
$\cdots$ y_{t-1} y_t y_{t+1} $\cdots$
$\overleftarrow{h}_{t-1}$ $\overleftarrow{h}_{t}$ $\overleftarrow{h}_{t+1}$
$\overrightarrow{h}_{t-1}$ $\overrightarrow{h}_{t}$ $\overrightarrow{h}_{t+1}$
$\cdots$ x_{t-1} x_t x_{t+1} $\cdots$

Figure 2.5: BiLSTM structure [19]

2.2 Conditional Random Field

Conditional Random Field (CRF) can be considered as a logarithmic linear model suitable for sequence tagging when context information is taken into consideration. Suppose there are two random variables X and Y, $P(Y|X)$ is the conditional probability distribution of Y given X, if Y can form a Markov random field, then the conditional probability distribution $P(Y|X)$ can be considered as a conditional random field, if X and Y have the similar structure, like $X = (X_1, X_2, \cdots, X_n)$, $Y = (Y_1, Y_2, \cdots, Y_n)$ are random variable sequences represented by linear chain, then $P(Y|X)$ is the linear chain conditional random field (linear-CRF). In sequence tagging tasks, CRF can define feature functions to capture the relations between labels and use these feature functions to give scores to different predicted tag sequence. In the model which implements NER task described by this thesis, we use linear-CRF as the layer on the top of Bi-LSTM layer.

2.3 Natural Language Processing

2.3.1 Bag of Words

Bag-of-Words (BoW) first appeared in NLP and Information Retrieval field. BoW generates a dictionary of all distinct words appear in all texts, and uses vectors to represent words or documents. The element in the vector is the number of appearances of every word in the dictionary. In this model, words in the texts appear independently, the sequence, semantics and syntax information is neglected. BoW can also be regarded as a histogram of frequency of words. However, when the dictionary becomes large, the number of dimension of the vector will increase large as well, which could bring more complexity in subsequent operations.

2.3.2 Word Embedding

Word Embedding is a method which maps words or phrases in vocabulary to vectors are consists of real numbers. The basic word embedding is *One-Hot Encoding* based on BoW. It uses a n-dimension vector to represent a word, n is the number of words in the vocabulary. In this word vector, except the value of k^{th} dimension is 1, values of other dimensions are 0, k is the sequence of this word in the vocabulary. Though One-Hot is easy to use and implement, it suffers from the curse of dimension when the vocabulary is too large. It does not consider the sequence of words in a document.

Another word embedding method is *Concurrence Matrix*. The idea is that it takes a word holds close relations with nearby words. It allows users to set a window size n which includes the target word and its nearby words (total $2n + 1$ words in the current window) to generates a concurrence matrix which has the dimension $D \times D$, D is the number of words in the vocabulary. To get the word vector, the number of concurrences should be calculated for each pair of words in the window, each column or row of the concurrence matrix is the vectorized representation for the corresponding word. Although this method considers the positions of words, it still yields too long word vectors.

Word2Vec

One of the most frequently used methods of word embedding is Word2Vec [20, 21] which is based on neural networks which can map words to low-dimensional space. Word2Vec is based on two neural network models: the *Continuous Bag-of-Words Model (CBOW)* and the *The Skip-Gram Model*. For example, in figure 2.6, the center word is 'hiking', the window size is 2. If Skip-Gram model has been used, it aims to generate each word in the context given center word, which means the model would learned by predicting words 'to', 'go', 'on', and 'Sunday' from word 'hiking'. While the CBOW model trys to learn by predicting 'hiking' by 'to', 'go', 'on', and 'Sunday'. The word embedding produced by Word2Vec can learn some certain semantic patterns because it uses contexts words to train. Due to its ability to evaluate the semantic similarities and make semantic analogies (like "King" - "Queen" $\approx$ "man" - "woman"") between words, Word2Vec becomes very popular.

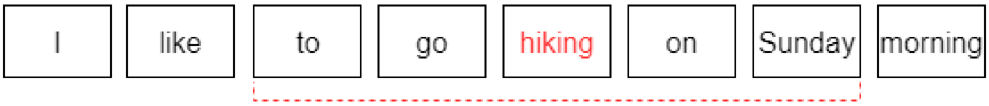


Figure 2.6: An example of context window for the word 'hiking'

Glove

After works of Word2Vec published, Pennington, Socher, and Manning [22] tried to make use of statistical information of the corpus and proposed another word embedding method called Glove. Based on the observation that two words are more relevant if they have shorter semantic distance, Glove uses word-pair co-occurrence to make related words more distinct. For example, there are three words i, j, and k, if word i and word j are relevant, while word i and word k are irrelevant, the ratio of co-occurrence $P(j|i)/P(k|i)$ should have a comparatively large value, if word pairs i, j and i, k are not relevant or relevant at the same time, the ratio is close to 1. So that this ratio could differentiate relevant words and irrelevant words, and get relevant word-pair. Glove is a count-based method, it can be regarded as a method to decrease the dimension of the co-occurrence matrix. There is not much difference on performance between Glove and Word2Vec, but Glove is more appropriate for tasks with a large volume of data because it is faster on parallelization. In this thesis, we use pre-trained Glove word vectors to initiate the embedding layer of the Bi-LSTM-CRF model for NER.

2.3.3 Named Entity Recognition

Named Entity Recognition (NER) is a classic NLP task, it aims to find pre-defined categories for each word in the text, these categories include names of persons, names of organizations, expressions of times, etc. The output of named entity tags could be used in many applications, such as find relevant web pages in the search engine, and posting ads according to identified texts. Most NER system takes a block of text without annotation, and outputs the annotated text or the sequence consists of name entity tags. For example, if the input is:

Bob bought a laptop from Canada.

Then the output should be:

B-PER O O O O B-LOC

In this example, 'B-PER' represents the beginning of persons' name, and 'B-LOC' means the beginning of a location's name, while 'O' indicates ordinary word which is not a named entity.

NER is a challenging problem, not only because labeled data is not enough in every language, but also lies in there are a few constraints on which kind of words can be named entities. Most existing methods to solve this task are linear statistical models, like Hidden Markov Models (HMMs), Maximum Entropy Markov Models (MEMMs) [23], and CRF [24]. With the thrift of deep learning, Convolutional Neural Network has been used to tackle NER problem [25], as well as RNN, like Graves and Schmidhuber [26] and Kawakami [27] used bi-directional RNN to do the sequence labelling. Currently the popular solution for NER is to use Bi-LSTM combined with CRF [28]. The Bi-LSTM layer could capture relations of texts from both directions of the text, and CRF layer is able to create rules of output labels to avoid situations like 'B-PER' followed by 'B-ORG' (it is not possible that the beginning of an organization's name follows the beginning of a person's name). This method can attain 90.94% accuracy according to Lample et al. [29].

In this thesis, we also use this Bi-LSTM combined with CRF, and try to use explainable deep learning methods to interpret this model for better debugging and improving the model in practical work.

2.4 Deep Learning for NLP

In recent years, deep neural networks have get considerable performance among many NLP tasks like sentiment analysis [30, 31, 32], syntactic analysis [33, 34], and machine translation [35, 36]. Different deep neural networks are applied for various tasks in NLP field. For example, in tasks of dialogue system, Zhou et al. [37] used the LSTM encoder on the top of CNN for multi-turn human-computer conversation which takes utterance into consideration to catch utterance-level relations in the text. For POS tagging, Andor et al. [38] proposed a transition-based approach combined with feed-forward neural network, and Huang, Xu, and Yu [28] also used Bi-LSTM to predict POS tags. Various deep models have become the new state-of-art methods in NLP field. Correspondingly there appears some explainable deep models focus on NLP tasks or adapted existed explainable methods to NLP tasks, related works will be introduced more in section 2.5.5.

2.5 Understanding Neural Networks

In this section, the concept of Explainable Artificial Intelligence and different kinds of explainable methods are discussed. The first type talks about models which are easy to understand by humans. Then we discuss more on explanations for complex models on specific instances (locally) or on the whole behaviors (globally). Some related works about applications in NLP field with explainable deep learning methods are introduced as well.

2.5.1 Explainability of Machine Learning

When the expert system appears [39, 40, 41], many researchers started to explore how to explain the intelligent system, which mainly focused on designing more explainable representations to reduce the complexity from complicated rules. However, AI systems nowadays are different from those rule-based systems in the past. The model becomes more complex because of the increased number of parameters and operations, thus it is more difficult to explain the decision of the model. There is no commonly agreed definition so far. Doshi-Velez and Kim [42] consider the interpretability (or explainability) as the ability to explain or to present in understandable terms to a human. Another elegant definition of interpretability from Miller [43] states that interpretability is the degree to which a human can understand the cause of a decision. Doran, Schulz, and Besold [44] proposed a new concept of interpretability, which is that an AI system is able to explain the decision-making process of a model using human-understandable features of the input data. Usually, simple models like linear classifier are easy to interpret, while a complicated model like a deep neural network is difficult to understand owing to its layer-wise structure and nonlinearity of computation. In this thesis, we define the explainability of an AI system as the ability to explain the reason why it makes the decision in a human-understandable way, and describe more about the explainable model for deep neural networks.

There comes an intuitive question. Why do we need explainability? According to Samek, Wiegand, and Müller [45], reasons can be described from four aspects: trust from users, modification of the system, learn from the system, and moral and legal issues.

- **Trust from users**. The complicated machine learning model is a black box. Normal users do not know how it makes decisions and how to make sure its decisions are correct. They may suspect the model and worry about if the model is trustable. Once the model can explain itself, users can know the rationale of the model, why it makes right or wrong decisions, so that users can have more trust to the system.

- **Modification of the system**. Some complex models like deep neural networks have too many operations and parameters, so they are difficult to explain. If developers do not know why it performs bad, then it is difficult to debug. Therefore, it should be easier to improve the AI system if the developer can easily modify it.

- **Learn from the system**. Sometimes we can learn new insights from the AI system. For example, AI systems outperform human Go players and perform strategies unexpected by human. We can extract some new knowledge or insights from the AI and explainable AI models can let us learn from it.

- **Moral and legal issues**. Though AI affects our daily life gradually, related legal issues such as how to assign the responsibility when AI makes the wrong decision, have not received much attention until recent years. It is difficult to find perfect solutions for these legal issues because we rely on black-box models.

In some situation, explainability is not necessary, such as applications in a certain domain is resistant to unexpected errors, or the domain which has been studied thoroughly and its application has been tested well, then the explainability for the model is unlikely to be a prerequisite.

2.5.2 Interpretable Models

Interpretable models are considered to be easier to understand by human without explicit explanations. For example, rule-based methods such as decision trees [46] uses nodes and branches to represent its reasoning when making decisions. It is more intuitive and straightforward for human to trace from a leaf to the root to understand the decision. Considering the difficulties in constructing high-accuracy interpretable trees, Letham et al. [47] proposed the Bayesian Rule List introduced Bayesian framework into rule-based methods. But this kind of methods suffers from scalability problem. When the number of nodes and rules grows fastly, understanding the classifier as a whole is difficult. Except for rule-based models, linear models and k-nearest neighbors (kNN) models are also believed easy to interpret.

2.5.3 Local Explainable Methods

For a given input point, if the explainable method generates explanation in a small region around this given point for the model, we call it as a local explanation. In local explanation, certain inputs and their corresponding predictions are usually taken as examples to explain how the model behaves. We categorize the local explanation into two types, namely model-unaware explanations and model-aware explanations, depending on if the explanation is generated using parameters or structures directly from the model.

Model-unaware explanations

Most works about Model-unaware explanations often derive explanations based on sensitivity. For example, Simonyan, Vedaldi, and Zisserman [48] use the squared partial derivatives as the class saliency score of an image classifier with respect to a given input image, and highlight the most sensitivity part on this image which gives spatial support to the prediction. Similarly, Li et al. [49] compute the first-order derivative as the salience score of a unit from different RNN classifiers for sentiment analysis, and generate heatmap matrices as explanations to which dimension in the embedding vector or which word contribute most to the prediction. Although generating explanations based on sensitivity is effective and intuitive, the high non-linearity of complex models may cause the explanation noisy.

Except for the sensitivity-based method, some works try to use a simple model to imitate the behavior of a complex model. Ribeiro, Singh, and Guestrin [50] proposed Local Interpretable Model-Agnostic Explanations (LIME), which is a method tries to apply a simple linear model to approximate the behavior of a complex model on a region around a single instance. This method takes one instance and permutes it to generate a new dataset and fit it with both complex model and simple model to see how the prediction changes. Based on this idea, Wu et al. [51] introduce a tree regularization method to make the decision boundaries of deep models to be easily understood by human. LIME-based methods are model-agnostic. Though this kind of methods has advantages like model flexibility and easy comparisons between different deep models, they have difficulties in understanding the global situation, and inconsistency among explanations of different samples[52].

Model-aware explanations

Methods derive model-aware explanations often make use of the parameters of the model. For CNNs, Selvaraju et al. [53] use the gradients from the final convolutional layer to visualize important pixels in the input images corresponding to the classification. Bach et al. [54] use Layer-wise Relevance Propagation (LRP) to flow the relevance score from the output layer to the input layer of CNNs with linear sum-pooling and convolution or simple multiple perceptron. This method could measure the contribution of each input variables. Hendricks et al. [55] implement an image classifier based on CNN which could output explanatory text by an RNN trained by descriptive labels and descriptions of the images. Kuwajima and Tanaka [56] proposed a general idea to give inference of decision for visual recognition tasks by extracting the overlaps between highly activated features and frequently activated features in each class. For RNNs, Murdoch and Szlam [57] propose an approach to generates representative phrases from the input for LSTM model, they validate these phrases are important features by using them to construct a simple rule-based model to approximate the performance of original LSTM model. Some works aim to visualize how the deep model learns, like Karpathy, Johnson, and Fei-Fei [58] visualize the hidden units of the RNN and find some learning mechanisms of how an LSTM language model learn. Strobelt et al. [59] develop a framework which allows users to choose the input range and visualize the corresponding change in hidden state patterns.

2.5.4 Global Explainable Methods

If the explanation is generated for the whole input space or as an overview of how the model behaves, we regard it as global explanation. Similar to the category of local explanations, we also discuss global explainable methods from model-unaware and model-aware approaches.

Model-unaware explanations

There is not much work about how to generate model-unaware explanations globally in a model-agnostic way. Ribeiro, Singh, and Guestrin [50] make the global explanations by presenting a set of representative local explanations to users one at a time. This method is easy to fail when there is too much training data, and users cannot remember a lot representative local explanations to form a global view.

Model-aware explanations

Bau et al. [60] take the activation of each neuron for an image as the semantic segmentation of concepts represented by this image. Through this dissection process, they align neurons and human-understandable concepts to assess how well a concept is represented by a single unit. Based on the idea of disentangled patterns, Zhang et al. [61] use an explanatory graph to represent the knowledge hierarchy of a pre-trained CNN, which enables the logic-based rules as the representations of the inner logic of the CNN knowledge, so that the explanation could be more concise and meaningful.

2.5.5 Explainable Methods for NLP

In additional to explainable models for visual information classification or prediction, there are many methods developed for text data. Like LRP [54] has been transferred to deep models for text data [62]. This method was adapted to sentiment analysis task [63] and neural machine translation field [64] as well. The sensitivity-based method has also been applied to NLP field. After constructing saliency map for every dimension of a word vector in sentiment analysis task [49], Li, Monroe, and Jurafsky [65] propose a general methodology to measure the importance of different representations, like the input word vector dimension, the intermediate hidden unit or the input word, by erasing various representations. There are also methods which could generate rationales as inference for the decision of deep models, like Zhang, Marshall, and Wallace [66] train a sentence-level CNN for text classification by using rationale annotations, Lei, Barzilay, and Jaakkola [67] develop a model to extract phrases which are enough to make predictions from the input text as the rationale of the predictions in a sentiment classifier consists of two modules.

Chapter 3

Approaches

In this chapter, we describe the Bi-LSTM-CRF model for NER, and how to apply LRP [54] to explain the Bi-LSTM layer of the model, as well as the approach to visualize word vectors and measure the effectiveness of CRF layer. First, we introduce the model as a whole, then analyze the explainability of Bi-LSTM-CRF model for NER layer by layer.

3.1 Bi-LSTM-CRF Model for NER

Most of the methods mentioned in Chapter 2 aim to reveal the explainability for tasks with the single output for each input instance. But in NER problem, both the input and the output are the sequence. To make NER explainable, in this thesis, we implement a named entity recognizer by Bi-LSTM and CRF model as same as the approach from Huang, Xu, and Yu [28], and use LRP [54] to measure the contribution of inputs to the prediction inspired by work from Arras et al. [62] and Ding et al. [64], the former apply this method to LSTM and the latter adapt it to explain neural machine translation, which is a sequence prediction task. As for the visualization, we take the similar methodology from Li et al. [49], the difference is that rather than use the partial derivative to colorized the saliency matrices, we use the relevance score to construct the heat map. So that we can visualize how word vectors or hidden state units contributed to the generalization of the output sequence.

For the neural model to implement the named entity recognizer, between the input layer and the output layer, we use the pre-trained word vectors Glove[1] [22] to initialize the embedding layer, then follows a bi-directional LSTM layer and a CRF layer. The structure of the model is shown in Figure 3.1.

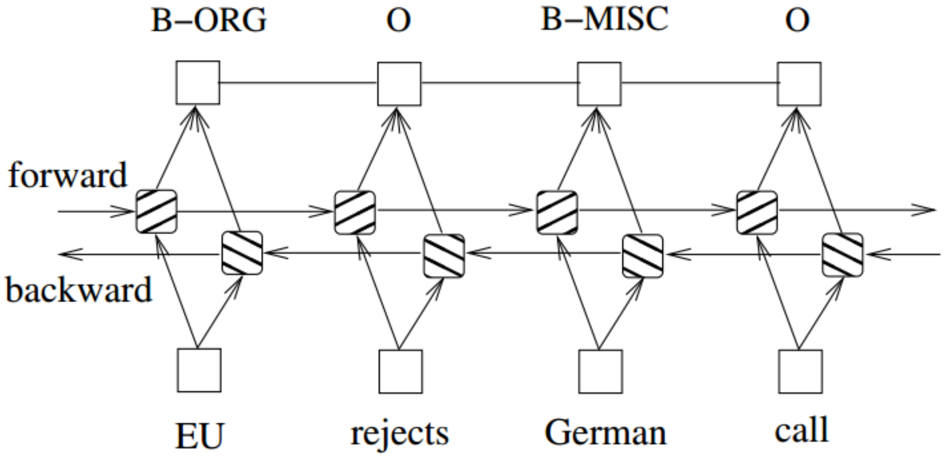


Figure 3.1: Bi-LSTM-CRF model for NER [28]

In this model, information from the past (via forward states) and the future (via backward states) can be used by the Bi-LSTM layer, while the CRF layer can capture the relation of contextual labels to assist the prediction of the current label. To clarity this model, we explain different layers in the model combined with the example stated in Figure 3.1.

In the input layer, the length of each input sequence will be padded to the same length. We suppose l is the length of each input sequence. The length of the input sequence is equal to the number of timesteps. At time t, the input layer takes the token of the current input sequence which could consist of one-hot-encoding word vectors or dense vectors. For example, if $t = 1$, the input sentence is "EU rejects German call", in this timestep the neural network takes the word representation of the word "EU". After l timesteps passed, the whole input sentence has been processed.

In the Bi-LSTM layer, each neuron consist of input gate, forget gate, and output gate as described in section 2.1.2. These neurons are devided into two directions, which takes each input sequence in order or in reverse order to process, so that the neural network can use both

[1] http://nlp.stanford.edu/data/wordvecs/glove.6B.zip

past and future information of current word. In every timestep, the model can output the hidden state of current timestep. But usually, we take the last hidden state as the output of a RNN in classification problems. Since NER is a sequence tagging problem, it predicts a tag for each components of the input sentence. Thus, the hidden state output in each timestep shows the predicted tag for current word. Considering the output of the Bi-LSTM layer in each timestep is a probability distribution of possible tags for current word, in this thesis, we use these hidden states before the last timestep, and get the highest entry of these vectors as the relevance score of the current predicted tag.

As for the CRF layer, we try to visualize how the CRF layer influence the prediction made by Bi-LSTM layer. Since we use linear chain CRF, we do not use LRP to explain the effect caused by this layer. By using the CRF layer after the Bi-LSTM layer, we can make use of information from neighbor tags to predict the current tag. A CRF layer includes a transition matrix as parameters which defines feature functions of CRF and capture the relation between different tags. For each input sentence, the CRF layer combined the predicted scores assigned to each word and the transition scores to choose the predicted tag sequence with highest score.

3.2 t-SNE for Embedding Visualization

In the training process, we use pre-trained Glove[22] word vectors to initialize the embedding layer. The reason we use pre-trained word embeddings is that it can get better performance than random embeddings or word vectors trained by limited tagging data. To capture the relations between different words vectors in the input text, we use t-Distributed Stochastic Neighbor Embedding (t-SNE)[68] to visualize these pre-trained word vectors, in order to show how similar words get together. t-SNE is a non-linear method which aims to reduce dimension of vectors with high dimensionality. To embeds high-dimensional data points to low-dimensional space, it turns the similarities of data points to probabilities, uses joint probability under Gaussian distribution to represent the similarity between data points in original high-dimensional space, while in low-dimensional space it uses t-distribution [69]. By calculating the Kullback-Leibler divergence of the joint probability of original space and embedded space, we can evaluate the performance of dimension reduction. t-SNE can keep similar words as close as possible while maximizing the distance among words are dissimilar.

3.3 Layer-wise Relevance Propagation for Bi-LSTM

We follow Bach et al. [54] to use LRP to compute relevance score of each hidden states or each variable in the input layer. Here we use an example shown in 3.2 to describe how LRP works in general.

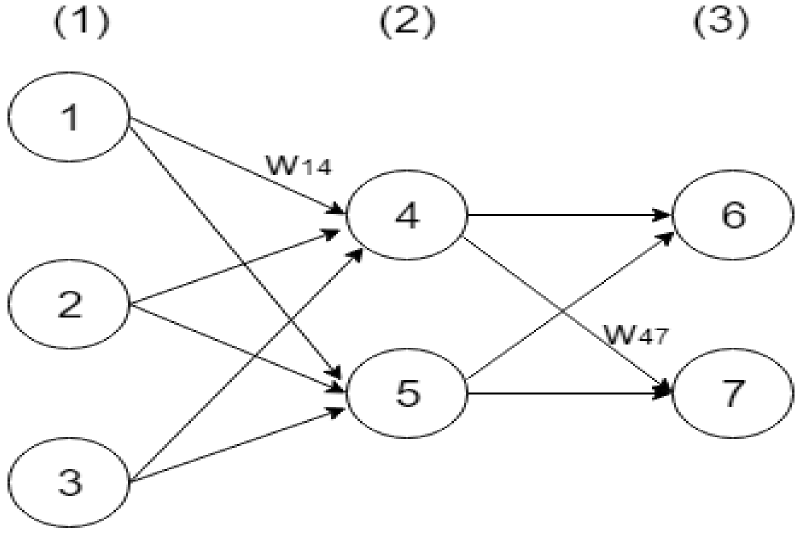


Figure 3.2: An example of how LRP works

LRP aims to redistribute the relevance score of activated neuron in the output layer to variables in the input layer via the backpropagation operation, the relevance score refers to the function value calculated by neuron networks of the activated unit in the output layer. In the following, we describe how LRP assign the relevance score to units in a neural model. There is a 3-layered NN shown in figure 3.2, the input layer has 3 neurons, while for the hidden layer and the output layer each has 2 units. Each layer and each unit are labeled as shown in the figure. In this example, we use $R_{i\leftarrow j}^{(l,m)}, (l < m)$ to represent the relevance distributed from the neuron j in layer m to the neuron i in layer l, $R_i^{(l)}$ refers to the relevance score of neuron i in layer l, a_i means the activation of neuron i, w_{ij} is the weight connection between neuron i and neuron j. For example, in figure 3.2, $R^{(1,2)}_{1\leftarrow 4}$ means the relevance score assigned from neuron 4 in layer (2) to the neuron 1 in layer (1), $R^{(1)}_{1}$ represents the relevance score assigned to neuron 1, w_{14} means the weight vector between neuron 1 and neuron 4.Then we illustrate how LRP calculates the relevance of input variables by the following procedure.

1. LRP propagates the relevance from units in output layer to units in intermediate layer. Here we calculate relevance propagate from neuron 6 to neuron 4 and neuron 5:

$$R^{(2,3)}_{4\leftarrow 6} = \frac{a_4 \cdot w_{46}}{a_4 \cdot w_{46} + a_5 \cdot w_{56}} \cdot R^{(3)}_{6} \tag{3.1}$$

$$R^{(2,3)}_{5\leftarrow 6} = \frac{a_5 \cdot w_{56}}{a_4 \cdot w_{46} + a_5 \cdot w_{56}} \cdot R^{(3)}_{6} \tag{3.2}$$

   Here $R^{(3)}_{6}$ refers to the function value calculated by the neural model, which means the prediction of neuron 6. The relevance from neuron 7 to neuron 1 can be calculated similarly as:

$$R^{(2,3)}_{4\leftarrow 7} = \frac{a_4 \cdot w_{47}}{a_4 \cdot w_{47} + a_5 \cdot w_{57}} \cdot R^{(3)}_{7} \tag{3.3}$$

$$R^{(2,3)}_{5\leftarrow 7} = \frac{a_5 \cdot w_{57}}{a_4 \cdot w_{47} + a_5 \cdot w_{57}} \cdot R^{(3)}_{7} \tag{3.4}$$

2. Then further propagate relevance $R^{(3)}_{6}$ to neurons in the input layer:

$$\begin{aligned} R^{(1,3)}_{1\leftarrow 6} =& \frac{a_1 \cdot w_{14}}{a_1 \cdot w_{14} + a_2 \cdot w_{24} + a_3 \cdot w_{34}} \cdot R^{(2,3)}_{4\leftarrow 6} + \\ & \frac{a_1 \cdot w_{15}}{a_1 \cdot w_{15} + a_2 \cdot w_{25} + a_3 \cdot w_{35}} \cdot R^{(2,3)}_{5\leftarrow 6} \end{aligned} \tag{3.5}$$

$$\begin{aligned} R^{(1,3)}_{1\leftarrow 7} =& \frac{a_1 \cdot w_{14}}{a_1 \cdot w_{14} + a_2 \cdot w_{24} + a_3 \cdot w_{34}} \cdot R^{(2,3)}_{4\leftarrow 7} + \\ & \frac{a_1 \cdot w_{15}}{a_1 \cdot w_{15} + a_2 \cdot w_{25} + a_3 \cdot w_{35}} \cdot R^{(2,3)}_{5\leftarrow 7} \end{aligned} \tag{3.6}$$

   We can use similar approach to get $R^{(1,3)}_{2\leftarrow 6}$, $R^{(1,3)}_{3\leftarrow 6}$, $R^{(1,3)}_{2\leftarrow 7}$, and $R^{(1,3)}_{3\leftarrow 7}$.

3. Then we can summarize the relevance of each neuron in the input layer obtained from neurons in output layer, which can be

calculated as:

$$\begin{aligned} R_1^{(1)} =& R_{1\leftarrow 6}^{(1,3)} + R_{1\leftarrow 7}^{(1,3)} \\ =& \frac{a_1 \cdot w_{14}}{a_1 \cdot w_{14} + a_2 \cdot w_{24} + a_3 \cdot w_{34}} \cdot (R_{4\leftarrow 6}^{(2,3)} + R_{4\leftarrow 7}^{(2,3)}) + \\ & \frac{a_1 \cdot w_{15}}{a_1 \cdot w_{15} + a_2 \cdot w_{25} + a_3 \cdot w_{35}} \cdot (R_{5\leftarrow 6}^{(2,3)} + R_{5\leftarrow 7}^{(2,3)}) \end{aligned} \tag{3.7}$$

We can also calculate $R_2^{(1)}$ and $R_3^{(1)}$ in similar way.

There is an assumption LRP based on: the relevance in every layer of the neural model keeps same. Take the model in figure 3.2 as instance, the relevance of all units in the neural network should satisfy:

$$R_6^{(3)} + R_7^{(3)} = R_4^{(2)} + R_5^{(2)} = R_1^{(1)} + R_2^{(1)} + R_3^{(1)} \tag{3.8}$$

Thus, if we calculate the relevance of neuron 1 the feed-forward prospective, same result can be obtained, we should get the same result as 3.7:

$$\begin{aligned} R_1^{(1)} &= R_{1\leftarrow 4}^{(1,2)} + R_{1\leftarrow 5}^{(1,2)} \\ &= \frac{a_1 \cdot w_{14}}{a_1 \cdot w_{14} + a_2 \cdot w_{24} + a_3 \cdot w_{34}} \cdot R_4^{(2)} \\ &+ \frac{a_1 \cdot w_{15}}{a_1 \cdot w_{15} + a_2 \cdot w_{25} + a_3 \cdot w_{35}} \cdot R_5^{(2)} \end{aligned} \tag{3.9}$$

From the forward propagation, we have:

$$\begin{aligned} R_4^{(2)} &= R_{4\leftarrow 6}^{(2,3)} + R_{4\leftarrow 7}^{(2,3)} \\ R_5^{(2)} &= R_{5\leftarrow 6}^{(2,3)} + R_{5\leftarrow 7}^{(2,3)} \end{aligned} \tag{3.10}$$

Take equation 3.10 into 3.9, we can get the result of 3.9 which is as same as the result from 3.7.

Now we can make a general rule of how LRP works. Suppose R_k is the relevance score of neuron k in layer $l+1$ in a multilayer neural network, R_j is the relevance score of neuron j in layer l. We can follow equation 3.11 to calculate R_j as:

$$R_j = \sum_k \frac{x_j w_{jk}}{\sum_j x_j w_{jk} + \epsilon} R_k \tag{3.11}$$

Where x_j is the activation of neurons at layer l, w_{jk} is the weight connection between neuron j and neuron k, and ϵ is a stabilization term to avoid division by zero.

As we can observe from Bi-LSTM, for NER problem, the output of each time step of Bi-LSTM is the prediction of current input word. So we take the score predicted by the Bi-LSTM layer as the initial weight for LRP. The goal of applying LRP is to visualize the hidden states generated from the input sequences by the model.

In RNNs, the approach to calculate the relevance of weight connections from the neuron in upper-layer to neuron in lower-layer of LSTMs is very similar to the standard LRP aforementioned, except a stabilizer ϵ (set to 0.001 in experiments) is added to the numerator of equation 3.11. Here we assume a part of the relevance was "absorbed" by the biases so conservation of relevance propagation is only kept approximately. We use $R_{j\leftarrow k}$ in equation 3.12 to imply the relevance received by neuron j in layer l from neuron k in layer $l+1$, where $x_k = \sum_j x_j \cdot w_{jk} + b_k$, therefore $\sum_k R_{j\leftarrow k}$ represent the relevance score of neuron j.

$$R_{j\leftarrow k} = \frac{x_j \cdot w_{jk} + \frac{\epsilon \cdot sign(x_k)}{N}}{x_k + \epsilon \cdot sign(x_k)} \cdot R_k \tag{3.12}$$

where $sign(x_j) = (1_{x_j \geq 0} - 1_{x_j \leq 0})$, x_j is the activation of neuron j, N is the number of all neurons in lower-layer which connected by j.

To quantify how much one contextual word vector contributes to a hidden state, equation 3.13 is introduced, where u is the word vector, v is the current hidden state.

$$R_{u\leftarrow v} = \sum_u \sum_v r_{u\leftarrow v} \tag{3.13}$$

Based on formulas mentioned above, we use algorithm 1 to illustrate the approach to visualize the impact generated by hidden states and word vectors to the prediction.

3.4 Explanation for the CRF Layer

In each timestep, the Bi-LSTM layer takes a word vector and output a vector which represents the probability distribution of possible tags for this word. The CRF layer takes the score of most possible tag for the current word as the emission score of this word. Then it adds the emission score and a transition score which represents the probability of a tag followed by another tag. The transition score describes the probability of a label followed by another label. For each input

Algorithm 1: Layer-wise relevance propagation for Bi-LSTM in Named Entity Recognition

```
Input: A pre-trained Bi-LSTM $G$ for a paired sequence
Output: Word-level relevance vector $R_w$
for unit $z$ in the output layer of $G$ do
    $R_z = f(z)$;
end
for unit $i$ in intermediate layer $l$ do
    $R_{i \leftarrow j}^{(l,l+1)} = \frac{x_i \cdot w_{ij} + \frac{\epsilon \cdot sign(x_j)}{N}}{x_j + \epsilon \cdot sign(x_j)} \cdot R_j$ ;
    $R_i^l = \sum_j R_{i \leftarrow j}$;
end
for word representation $w$ in input sequence do
    $R_w = \sum_i R_i^l$;
end
```

sentence, the CRF layer calculates the corresponding tag sequence by taking emission scores, transition score, and previous tags into consideration, and choose the sequence which gets the highest score. The entries of the transition score matrix are parameters of the model which can be trained. To explain the influence from the CRF layer, we used the weights and biases to calculate the emission score vector for each word, and compare it with the output from the CRF layer, so that we can observe how transition score matrix change the prediction from Bi-LSTM layer. Considering the parameters of CRF layer are trained together with parameter on other layers, so we just apply LRP to Bi-LSTM layer in a pre-trained Bi-LSTM model without CRF layer, while when generate explanation of CRF layer, we use the Bi-LSTM-CRF model as a whole.

Chapter 4

Experiment and Result Analysis

4.1 Dataset

In the experiments of the named entity recognizer implemented by Bi-LSTM CRF model in this thesis, we use the CoNLL 2003 English named entity dataset [70], which is a public dataset contains independent named entity tags. Table 4.1 shows the size of sentences and tokens in training, validating and test datasets.

English data	Sentences	Tokens
Training	14,987	203,621
Validating	3,466	51,362
Test	3,864	46,435

Table 4.1: Number of sentences and tokens in each data file

This dataset contains four different types of named entities: locations (LOC), persons (PER), organizations (ORG) and miscellaneous names (MISC). Each line of the data file is consists of two fields, namely a word and its named entity. In each type of named entities, the first word of a named entity is tagged ’B-XXX’ to show this word is the beginning of a named entity, the rest words of this named entity are tagged as ’I-XXX’. Words tagged with ’O’ means they are outside of named entities. Table 4.2 shows the number of named entities in data file.

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

Table 4.2: Number of each named entity

4.2 Environment and Parameters

In this thesis project, experiments were run on a machine with following configurations:

- Intel Core i7-6500U CPU, 2.5 GHz, 2 cores
- 8 GB DDR4 RAM
- Windows 10 64-bit OS

The Bi-LSTM-CRF model used in this thesis is implemented by Keras and Python. Keras is an open source project which provides high-level API to implement deep neural networks and runs on top of deep learning libraries like Tensorflow and Theano. It can make the information about other libraries or tools used in this thesis is listed below in table 4.3.

Library	Version
Python	3.5.4
Keras	2.1.5
TensorFlow	1.5.0
Sickit-learn	0.19.1
Pandas	0.22.0
numpy	1.14.2
matplotlib	2.2.2

Table 4.3: Tools and libraries used in this thesis

In the experiment, we employed one hidden-layer bi-directional LSTM, which takes a sequence of words $x_1, x_2, \cdots, x_T$ and its reversed order format as input, each word vector is represented by a word embedding has 100 dimensions. The size of nodes in the hidden layer is 100 as well, and the output is a sequence of named entity tags, each tag is chosen from 9 tags, namely B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, B-MISC, I-MISC, O. The output sequence has the same size as the input sequence. We train the Bi-LSTM-CRF model with batch size 32 and 5 epochs. Another thing to note is that we use LRP for Bi-LSTM model without CRF layer, because in this way the heat map can show how the Bi-LSTM layer make correct predictions, which can be easier to understand. However, in Bi-LSTM-CRF model, all parameters are trained simultaneously, it could cause some confusion from the result of Bi-LSTM layer.

4.3 Result Analysis

4.3.1 Visualizing Word Embeddings by t-SNE

We apply t-SNE to visualize the pre-trained embedding vectors as shown in Figure 4.1. This figure shows the visualization of all word vectors mapped to 2-dimensional space.

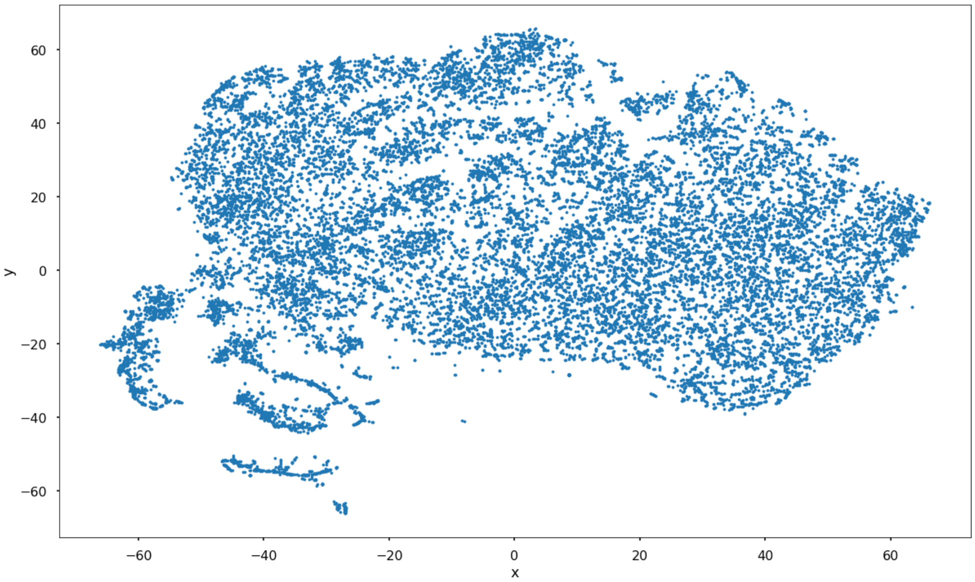


Figure 4.1: Glove embeddings for training data

Zoom in the figure above, we can visualize how words close to each other in a certain region as shown in figure 4.2. Two axes represent the position in each direction for the word vectors after decrease the dimension. For example, in figure 4.2, these words which are close to each other represent names of countries or cities, they are similar in way of use, which meets the idea of word embedding.

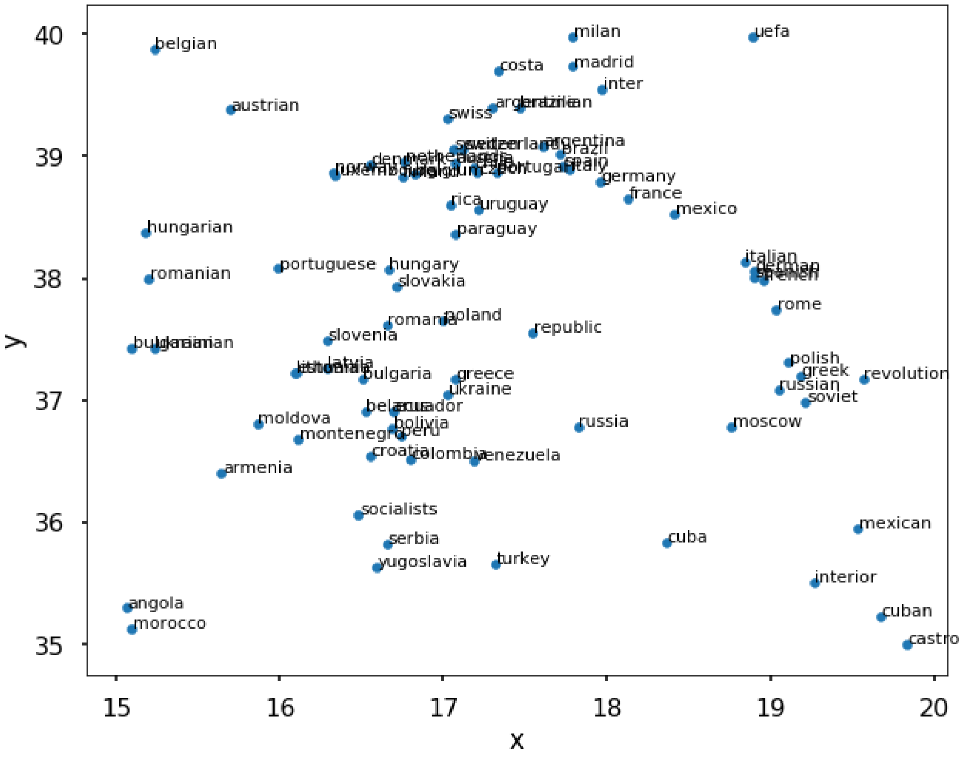


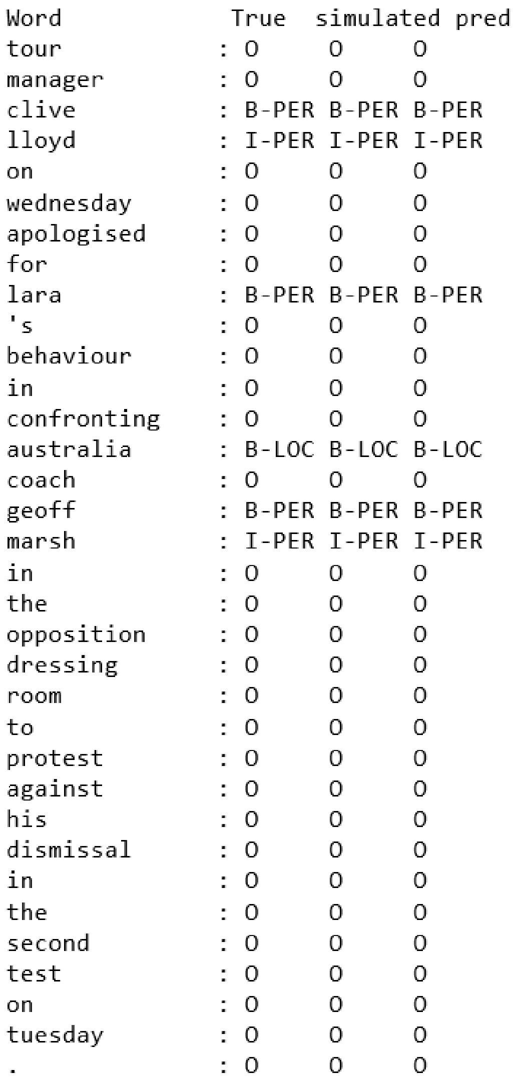
Figure 4.2: Local word embeddings represent names of countries

4.3.2 Visualization of the Bi-LSTM Layer

Considering LRP is a local explainable method, it generates explanations for certain input instance, so here are a few examples to show how it visualize the importance of single words in the input sequence.

Visualizing contribution of contextual words in hidden states

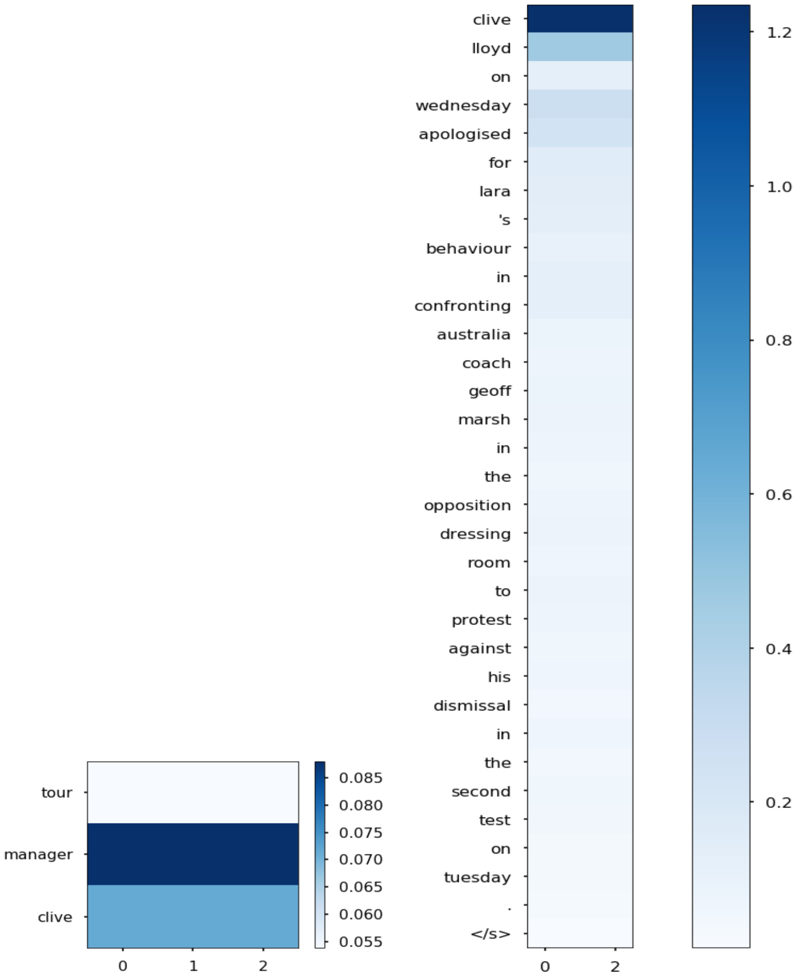
As shown in figure 4.3, the input sequence, and the predicted tag sequence are listed. The 'simulated' column shows the result calculated by all parameters of Bi-LSTM from the forward phase in LRP method. The 'True' column represents real tags of the input sequence, while the 'pred' column shows the predictions generated by the model. The 'simulated' column should be as same as the 'pred' column because it is generated via a forward propagate process applied with all parameters generated by the model.



```
Word           True  simulated pred
tour          : O     O     O
manager       : O     O     O
clive         : B-PER B-PER B-PER
lloyd         : I-PER I-PER I-PER
on            : O     O     O
wednesday     : O     O     O
apologised    : O     O     O
for           : O     O     O
lara          : B-PER B-PER B-PER
's            : O     O     O
behaviour     : O     O     O
in            : O     O     O
confronting   : O     O     O
australia     : B-LOC B-LOC B-LOC
coach         : O     O     O
geoff         : B-PER B-PER B-PER
marsh         : I-PER I-PER I-PER
in            : O     O     O
the           : O     O     O
opposition    : O     O     O
dressing      : O     O     O
room          : O     O     O
to            : O     O     O
protest       : O     O     O
against       : O     O     O
his           : O     O     O
dismissal     : O     O     O
in            : O     O     O
the           : O     O     O
second        : O     O     O
test          : O     O     O
on            : O     O     O
tuesday       : O     O     O
.             : O     O     O
```

Figure 4.3: Prediction generated by Bi-LSTM layer

According to the prediction result of this sentence in figure 4.3, the named entity tag for the word "clive" is "B-PER" (the beginning of a person's name). We regard "clive" as a target word, and the rest words are regarded as contextual words. By applying the LRP method, we can obtain a matrix of relevance scores of every context word in forwarding and backward hidden states as shown in figure 4.4. In this figure, every grid represents the relevance level of each contextual word contribute to the prediction of the target word. 4.4(a) and 4.4(b) represent forward hidden states and backward hidden states respectively.

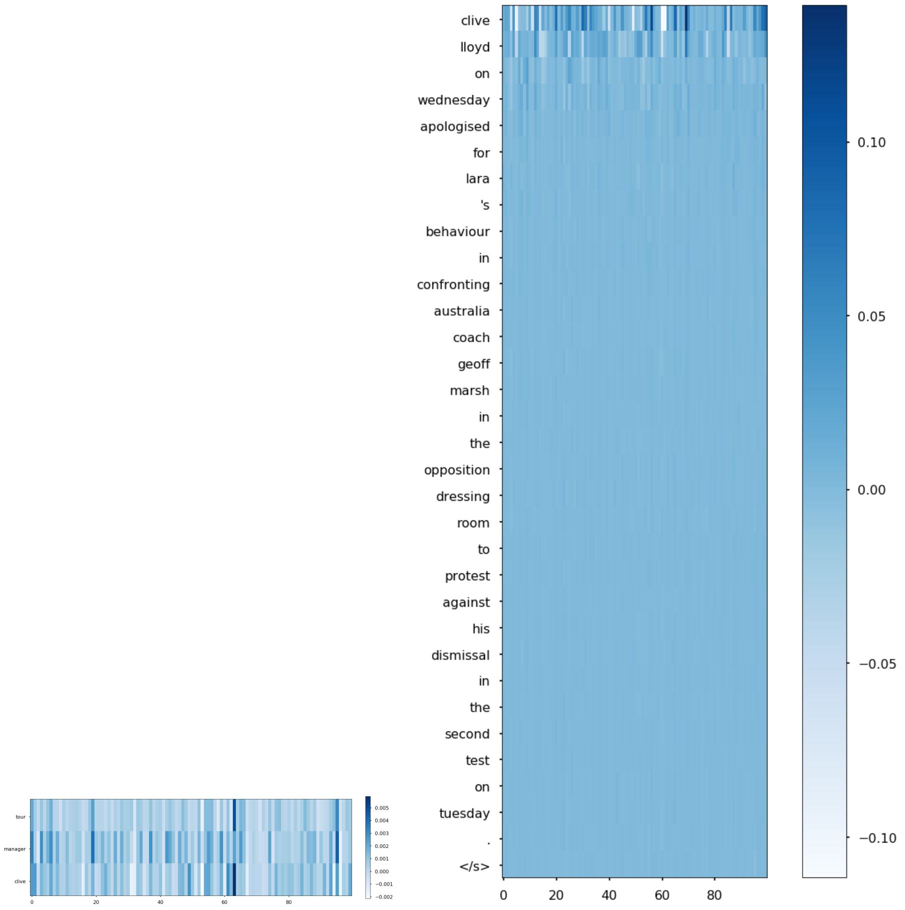


(a) Visualizing forward hidden states for a content word "clive"

(b) Visualizing backward hidden states for a content word "clive"

Figure 4.4: Heatmap of a content word "clive" in Bi-LSTM layer in two directions

From figure 4.4 we can observe that the word "manager" (the one before "clive") and the word "lloyd" (the one after "clive") contribute more to the hidden states of "clive" than other words, which means these two adjacent words shows more importance when the model is predicting the named entity tag for "clive". Considering the word before "clive" is used to describe a person's occupation, and the word after "clive" is the intermediate word of the person's name, also these two words are closest to the target word, it is easy to understand why the neural network thinks these two words contributes most to the prediction of the target word. The relevance of other contextual words decreases when the distance of them to the target word becomes larger. It is reasonable to say that the model captures the relation between nearby words and the target word in this instance. But from 4.4(b), it is clear that the relevance of the word "clive" is more concentrated on itself. To be more specific, we visualize how every unit in the Bi-LSTM layer contribute to the result of a certain word. In figure 4.5, the relevance score of every dimension of hidden states for predicting the tag



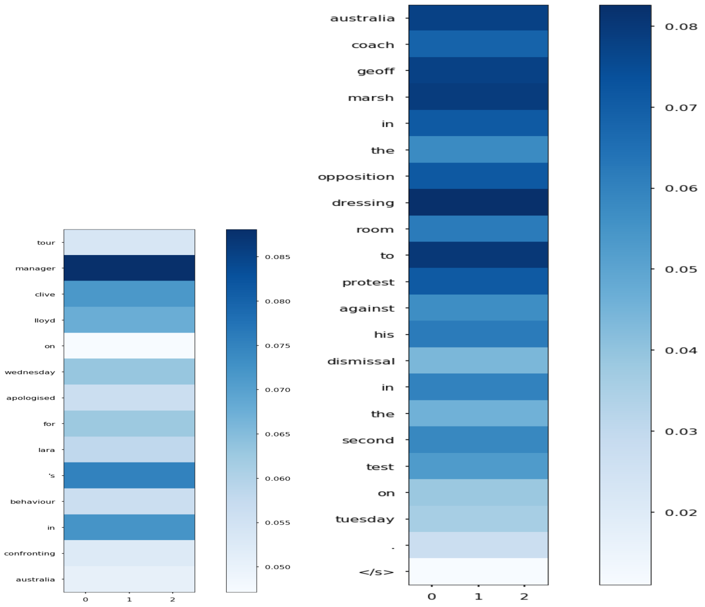
(a) Visualizing dimensions of forward hidden states for content word "clive"

(b) Visualizing dimensions backward hidden states for content word "clive"

Figure 4.5: Heatmap of every unit for a content word in Bi-LSTM layer in two directions

of the word "clive" is shown. It follows the trends in figure 4.4, namely nearby contextual words contributes more to the target word. But we notice there are some salient dimensions in both forward and backward order. From 4.5(a), the 63th unit of the word vectors represent "tour" and "clive" contribute more relevance, while in the backward state, the 69rd unit of the word representation of "clive" shows most importance. According to this visualization of every unit's relevance, we can find have a more direct impression of which units influence the prediction.

We can also check the contribution of contextual words to any target word. For example, from figure 4.3, we can notice that the predicted tag for the word "australia" is "B-LOC" (the beginning of a location). While from the heat map as shown in figure 4.6(a), the important words are "manager" and "dressing" (both are a bit far away from the target word), though "australia" itself shows quite a relevance. It is hard to explain why a location can be tagged as the named entity "B-LOC", we can only conclude that contextual words mentioned above give the most contribution to the prediction for this target word.



(a) Visualizing forward hidden states for a content word "australia"

(b) Visualizing backward hidden states for a content word "australia"

Figure 4.6: Heatmap of content word "australia" in Bi-LSTM layer in two directions

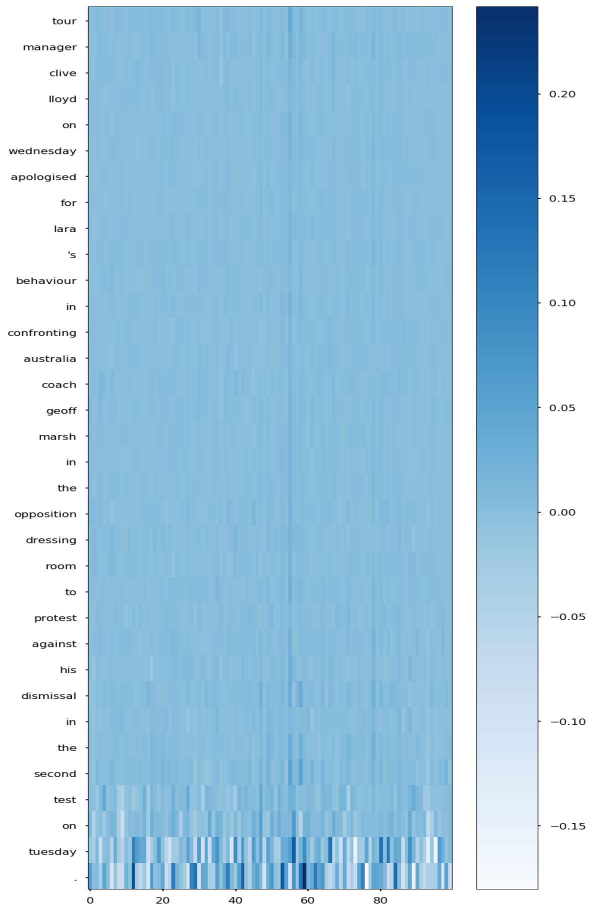
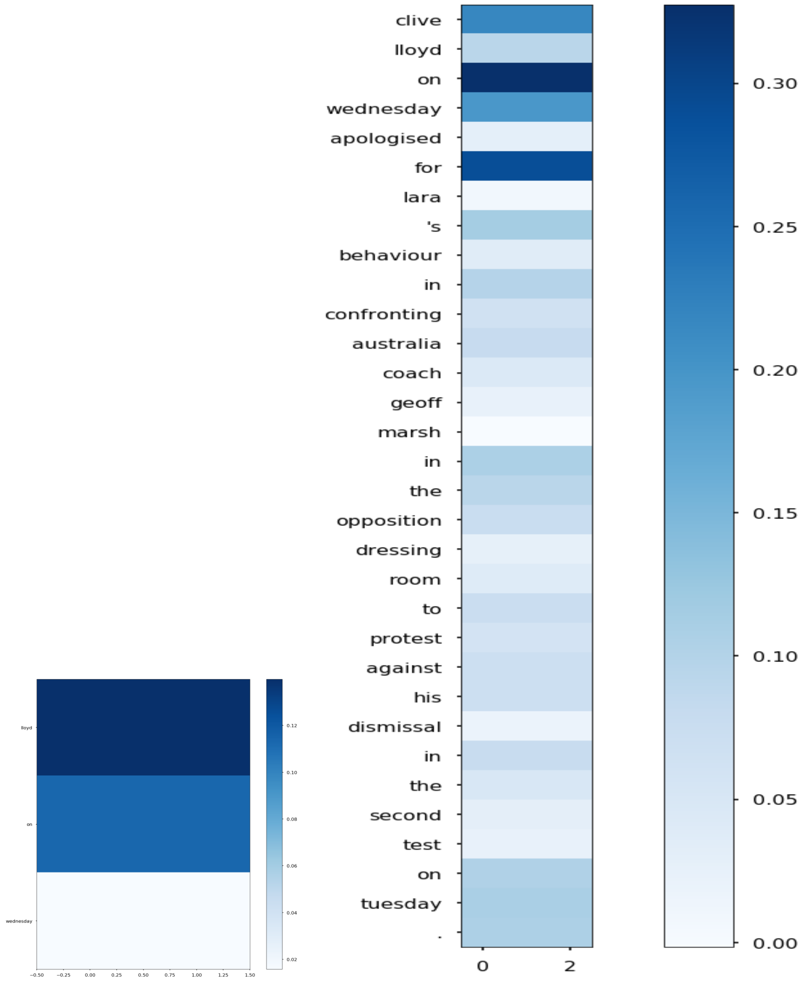


Figure 4.7: Visualize relevance of embedding layer for the last hidden state

Visualizing the relevance of contextual word vectors

Except for the relevance of hidden states, we also want to know which dimension of word vectors has more relevance to the prediction result. Figure 4.7 shows the relevance score of the word vectors contribute to the final hidden state for the input sentence aforementioned. Each row means the representation of a word, which is a 100-dimension vector. Each grid is the dimension of corresponding word vectors. As shown in figure 4.7, to contribute the last hidden state, namely the prediction for the last word, its nearby contextual word vectors shows more importance, and the 12th, 59th, and 84th dimensions perform more relevance in the generation of the tag of the last word. We can observe that the Bi-LSTM keep focus on contextual words near the last word.

For the case mentioned above, we can also visualize the relevance of the word vectors as shown in figure 4.8. We can observe that for the forward sequence, the word "tour" shows more relevance while in the backward sequence the word "on" shows most importance when predicting the tag of the target word "clive". Compare to figure 4.4, we can show how relevance transfers among units in different layers.

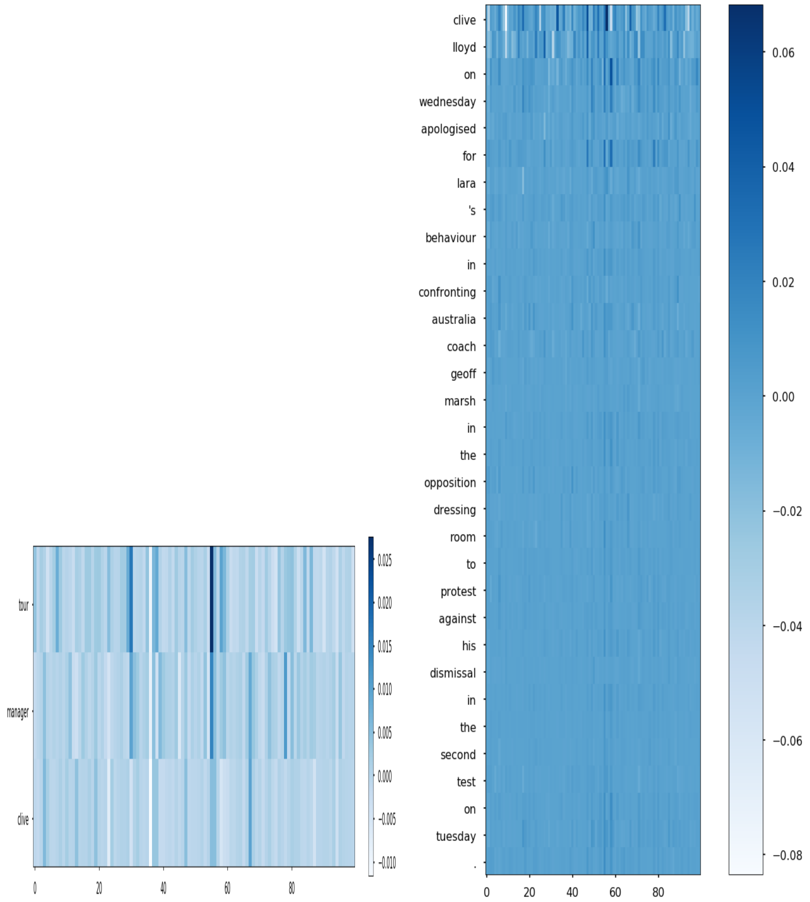


(a) Visualizing forward word vectors for word "clive"

(b) Visualizing backward word vectors for word "clive"

Figure 4.8: Visualizing word vectors of word "clive" in embedding layer in two directions

To be more clear, we visualize every dimension of word vectors from two directions in the embedding layer. From figure 4.9, the 55th dimension of the word "tour" and the 61st dimension of the word "clive" show more relevance than other words in the same direction.



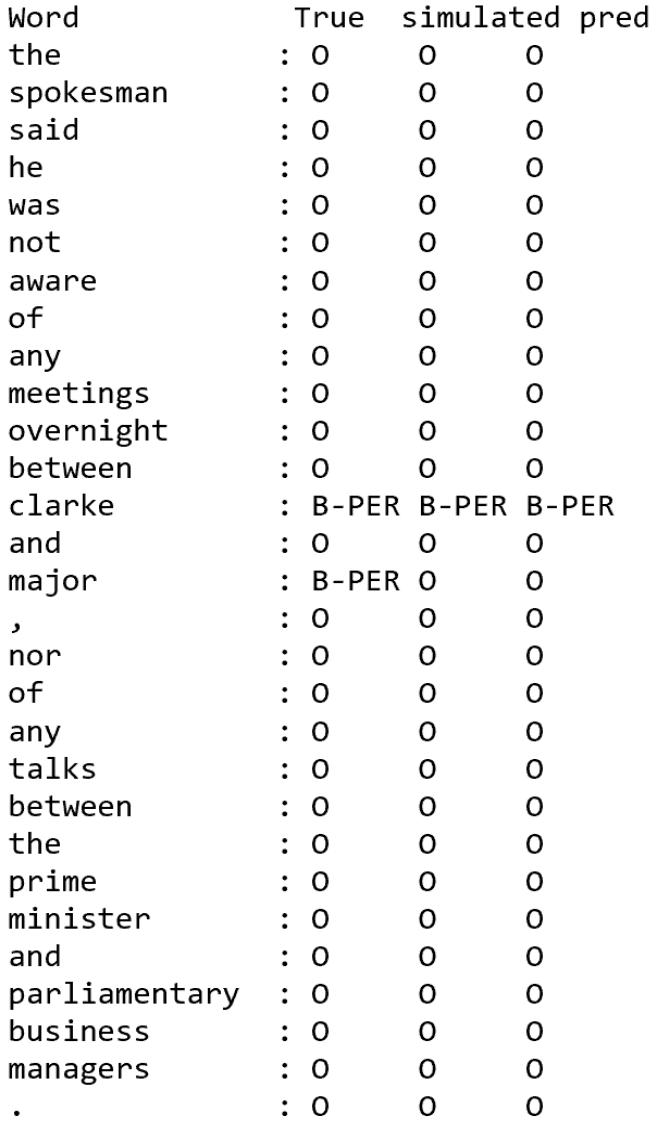
(a) Visualizing dimensions of forward word vectors for word "clive"

(b) Visualizing dimensions backward word vectors for word "clive"

Figure 4.9: Visualizing dimensions of the word vector for "clive" in embedding layer in two directions

Visualizing word relevance for wrong predictions

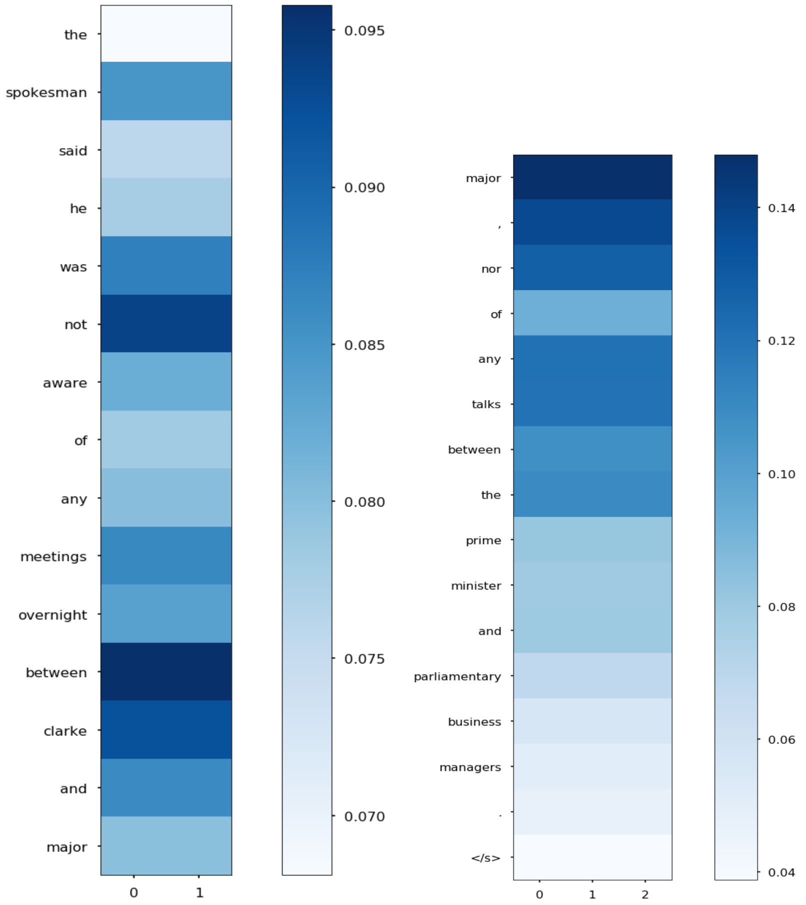
Another interesting problem is that how words contribute to wrong predictions. In figure 4.10, there is a predicted with wrong predicted tags, from that we can observe the tag of the word "major" is predicted wrongly. The true tag is ′B-PER′ while the prediction is ′O′. We visualize the forward hidden states and backward hidden states for this word "major" and take it as the target word as shown in figure 4.11. According to 4.11(a), in the forward hidden states of the word "major",



```
Word              True  simulated pred
the             : O     O     O
spokesman       : O     O     O
said            : O     O     O
he              : O     O     O
was             : O     O     O
not             : O     O     O
aware           : O     O     O
of              : O     O     O
any             : O     O     O
meetings        : O     O     O
overnight       : O     O     O
between         : O     O     O
clarke          : B-PER B-PER B-PER
and             : O     O     O
major           : B-PER O     O
,               : O     O     O
nor             : O     O     O
of              : O     O     O
any             : O     O     O
talks           : O     O     O
between         : O     O     O
the             : O     O     O
prime           : O     O     O
minister        : O     O     O
and             : O     O     O
parliamentary   : O     O     O
business        : O     O     O
managers        : O     O     O
.               : O     O     O
```

Figure 4.10: Wrong prediction generated by Bi-LSTM layer

words "not" and "between" shows more importance to the generation of the tag of the target word, and in the backward hidden states, the relevance focused on the target word itself. One possible reason could be the model does not capture the relation expressed by "between", though it predicts the word "clark" as ’B-PER’, which means the model does not notice the concatenation between the word "clark" and the target word, so it generates different tags for these two words.



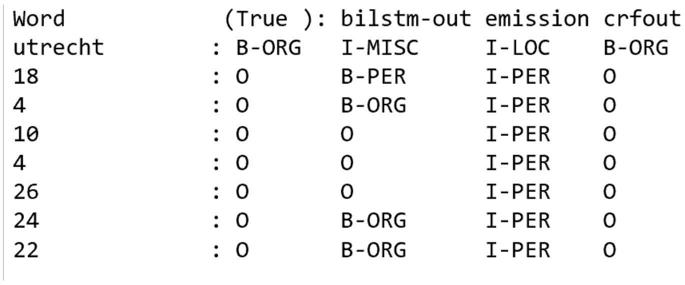
(a) Visualizing forward hidden states for a content word "major"

(b) Visualizing backward hidden states for a content word "major"

Figure 4.11: Visualizing hidden states of content word "major" in Bi-LSTM layer in two directions

4.3.3 Influence from the CRF Layer

From Keras we can get parameters from the CRF layer. By using the output from Bi-LSTM layer, we calculate the emission score vectors for each word in an input sentence, and listed these result in figure



```
Word           (True ): bilstm-out emission crfout
utrecht        : B-ORG   I-MISC     I-LOC    B-ORG
18             : O       B-PER      I-PER    O
4              : O       B-ORG      I-PER    O
10             : O       O          I-PER    O
4              : O       O          I-PER    O
26             : O       O          I-PER    O
24             : O       B-ORG      I-PER    O
22             : O       B-ORG      I-PER    O
```

Figure 4.12: Prediction from Bi-LSTM layer and emission vectors

4.12. From this figure, we can observe the difference between results from Bi-LSTM layer and final prediction generated by CRF layer, in order to know how CRF layer changes the results. We can also observe the influence from transition score matrix in CRF layer according to the difference between column ′emission′ and output from CRF layer (column ′crfout′). However, the result does not show some relations between results from the intermediate layer and the final output. Perhaps it is because the parameters of the CRF layer are trained together with other parameters rather than step by step. Therefore further investigation regards the effect of transition score matrix to every word should be made, but this is out of the scope of this thesis since it focuses more on the explanation of deep neural networks while CRF is a statistical modeling method. This could be a part of future work as mentioned in section 5.2.

Chapter 5

Discussion and Conclusion

5.1 Discussion

In this thesis, we applied different methods to make Bi-LSTM-CRF model explainable according to features of each layers. To understand the deep model used in the named entity recognizer, we visualize the behavior of this model. For example, t-SNE is used to visualize the relation of different word vectors. Considering the pre-trained Glove embedding has high dimensions for each word, we use t-SNE to reduce dimension of word vectors, map them to a 2-dimension coordinate system, and use the distance between coordinates to represent the similarity among words. Though t-SNE performs good on visualization, it can cause large memory usage and long running time. If there is a high-dimensional dataset and we do not know if it is separable, it is suitable to project it to low-dimensional space by t-SNE and check the separability of the dataset.

To visualize how the deep neural network behaves, sensitivity-based or saliency-based methods which measure the importance of each neuron could be useful. Though they are simple and intuitive, they suffer from noise generated by high non-linearity of complex models. In this thesis, we applied LRP to evaluate the importance of each unit in each layer of the neural network. Although LRP is a model-aware method, we make it adapted to NER problem. Now it can be used for multiple types of NN. From experiments in [45] and [62], LRP outperformed saliency-based method Sensitivity Analysis [48] because LRP showed better performance on recognizing units with most contributions for the prediction. Thus, if the deep model of the application is not known, saliency-based methods could be good choices. For model-agnostic applications, LIME is also applicable by perturbing inputs and observing how the black box model performs. If user wants to compare two machine learning models or to change models more frequently, LIME can perform well because its model-agnostic feature, though it has disadvantages like inconsistency among explanations of different samples. For the named entity recognizer implemented in this thesis project, we chose LRP to explain the Bi-LSTM layer of the model, not only because it is easier to be adapted for sequence tagging problem solved by RNN than LIME, but also it can presents how the relevance score flow inside the model by plotting heat maps, which meets the requirements of the development team from Seavus.

There are several weak points of this work, like the explanation for CRF layer is still not clear and it is hard to make inference about why a word is predicted to a named entity. How to use this method to analyze wrong predictions needs further consideration, as well as the the evaluation methods to make sure the explanation generated from LRP is correct.

5.2 Future Work

Though we applied the visualization method to show how the deep neural network "think" to some extent, the explanation should be more clear and easier to understand by normal users. Since we did not find much related works of applying explainable methods to NER model, we learn from ideas in related works which apply the LRP method to RNN and sequence tagging problem. Some challenges still need to be dealt with, like how to explain this model from an integrated view rather than layer-by-layer. We also need to figure out how to visualize the influence from CRF layer to the prediction. Further works to make this task more explainable includes how to apply other explainable methods to NER, and a more general approach to make sequence tagging problem solved by deep neural networks more interpretable.

5.3 Summary and Conclusion

In this thesis, we adapted LRP, an interpretable method for deep neural models, to make Bi-LSTM-CRF model explainable for NER task. We first introduce the motivation, and give a clear description of the goals of this thesis, then follows the background, which includes details of every component of the Bi-LSTM-CRF model. Considering the main purpose of this thesis is to implement an explainable named entity recognizer, some related works of explainable deep learning methods are introduced and categorized as local or global explainable methods. Then we describe several related works which talk about how to apply explainable methods in NLP tasks. Then approaches which used to implement the explainable NER are described thoroughly, each component of this model and methods to apply LRP to this model are depicted in detail. Finally, we visualize how much each word contribute to the output and how contextual words related, and give several examples to see how to explain the wrong prediction. Considering NER is an important task in NLP, the method we provide can give some insights about how to visualize deep neural networks used for NER. Since both the input and output of NER are the sequence, our approaches can also give some ideas of how to make the deep models explainable in sequence tagging problems as well.

Bibliography

[1] Michael J Pazzani and Daniel Billsus. “Content-based recommendation systems”. In: *The adaptive web*. Springer, 2007, pp. 325–341.

[2] Bayan Abu Shawar and Eric Atwell. “Machine Learning from dialogue corpora to generate chatbots”. In: *Expert Update journal* 6.3 (2003), pp. 25–29.

[3] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[4] Dayong Wang et al. “Deep learning for identifying metastatic breast cancer”. In: *arXiv preprint arXiv:1606.05718* (2016).

[5] David Gunning. “Explainable artificial intelligence (xai)”. In: *Defense Advanced Research Projects Agency (DARPA), nd Web* (2017).

[6] Anne Håkansson. “Portal of research methods and methodologies for research projects and degree projects”. In: *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*. CSREA Press USA. 2013, pp. 67–73.

[7] Arthur L Samuel. “Some studies in machine learning using the game of checkers”. In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[8] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[9] Erik Cambria and Bebo White. “Jumping NLP curves: A review of natural language processing research”. In: *IEEE Computational intelligence magazine* 9.2 (2014), pp. 48–57.

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

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[12] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), p. 484.

[13] William Chan et al. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 4960–4964.

[14] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[15] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[16] Christopher Olah. *Understanding LSTM Networks*. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed August 20, 2018.

[17] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[19] Peilu Wang et al. "A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding". In: *arXiv preprint arXiv:1511.00215* (2015).

[20] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[21] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[23] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. "Maximum Entropy Markov Models for Information Extraction and Segmentation." In: *Icml*. Vol. 17. 2000. 2000, pp. 591–598.

[24] John Lafferty, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: (2001).

[25] Ronan Collobert et al. "Natural language processing (almost) from scratch". In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537.

[26] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5-6 (2005), pp. 602–610.

[27] Kazuya Kawakami. "Supervised Sequence Labelling with Recurrent Neural Networks". PhD thesis. PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

[28] Zhiheng Huang, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991* (2015).

[29] Guillaume Lample et al. "Neural architectures for named entity recognition". In: *arXiv preprint arXiv:1603.01360* (2016).

[30] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[31] Mohit Iyyer et al. "Deep unordered composition rivals syntactic methods for text 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 1: Long Papers)*. Vol. 1. 2015, pp. 1681–1691.

[32] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[33] Danqi Chen and Christopher Manning. “A fast and accurate dependency parser using neural networks”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 740–750.

[34] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[35] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[36] Jacob Devlin et al. “Fast and robust neural network joint models for statistical machine translation”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2014, pp. 1370–1380.

[37] Xiangyang Zhou et al. “Multi-view response selection for human-computer conversation”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 372–381.

[38] Daniel Andor et al. “Globally normalized transition-based neural networks”. In: *arXiv preprint arXiv:1603.06042* (2016).

[39] William John Clancey. *The epistemology of a rule-based expert system: A framework for explanation.* Tech. rep. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1982.

[40] William Swartout, Cecile Paris, and Johanna Moore. “Explanations in knowledge systems: Design for explainable expert systems”. In: *IEEE Expert* 6.3 (1991), pp. 58–64.

[41] Robert Neches, William R. Swartout, and Johanna D. Moore. “Enhanced maintenance and explanation of expert systems through explicit models of their development”. In: *IEEE Transactions on Software Engineering* 11 (1985), pp. 1337–1351.

[42] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[43] Tim Miller. “Explanation in artificial intelligence: insights from the social sciences”. In: *arXiv preprint arXiv:1706.07269* (2017).

[44] Derek Doran, Sarah Schulz, and Tarek R Besold. “What does explainable AI really mean? A new conceptualization of perspectives”. In: *arXiv preprint arXiv:1710.00794* (2017).

[45] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. “Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models”. In: *arXiv preprint arXiv:1708.08296* (2017).

[46] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[47] Benjamin Letham et al. “Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model”. In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371.

[48] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[49] Jiwei Li et al. “Visualizing and understanding neural models in NLP”. In: *arXiv preprint arXiv:1506.01066* (2015).

[50] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Why should i trust you?: Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[51] Mike Wu et al. “Beyond sparsity: Tree regularization of deep models for interpretability”. In: *arXiv preprint arXiv:1711.06178* (2017).

[52] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Model-agnostic interpretability of machine learning”. In: *arXiv preprint arXiv:1606.05386* (2016).

[53] Ramprasaath R Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.” In: *ICCV*. 2017, pp. 618–626.

[54] Sebastian Bach et al. “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”. In: *PloS one* 10.7 (2015), e0130140.

[55] Lisa Anne Hendricks et al. “Generating visual explanations”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 3–19.

[56] Hiroshi Kuwajima and Masayuki Tanaka. “Network Analysis for Explanation”. In: *arXiv preprint arXiv:1712.02890* (2017).

[57] W James Murdoch and Arthur Szlam. “Automatic rule extraction from long short term memory networks”. In: *arXiv preprint arXiv:1702.02540* (2017).

[58] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[59] Hendrik Strobelt et al. “Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks”. In: *IEEE transactions on visualization and computer graphics* 24.1 (2018), pp. 667–676.

[60] David Bau et al. “Network dissection: Quantifying interpretability of deep visual representations”. In: *arXiv preprint arXiv:1704.05796* (2017).

[61] Quanshi Zhang et al. “Interpreting cnn knowledge via an explanatory graph”. In: *arXiv preprint arXiv:1708.01785* (2017).

[62] Leila Arras et al. “" What is relevant in a text document?": An interpretable machine learning approach”. In: *PloS one* 12.8 (2017), e0181142.

[63] Leila Arras et al. “Explaining recurrent neural network predictions in sentiment analysis”. In: *arXiv preprint arXiv:1706.07206* (2017).

[64] Yanzhuo Ding et al. “Visualizing and understanding neural machine translation”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 1150–1159.

[65] Jiwei Li, Will Monroe, and Dan Jurafsky. “Understanding neural networks through representation erasure”. In: *arXiv preprint arXiv:1612.08220* (2016).

[66] Ye Zhang, Iain Marshall, and Byron C Wallace. “Rationale-augmented convolutional neural networks for text classification”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2016. NIH Public Access. 2016, p. 795.

[67] Tao Lei, Regina Barzilay, and Tommi Jaakkola. “Rationalizing neural predictions”. In: *arXiv preprint arXiv:1606.04155* (2016).

[68] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[69] J Pfanzagl and O Sheynin. “Studies in the history of probability and statistics XLIV A forerunner of the t-distribution”. In: *Biometrika* (1996), pp. 891–898.

[70] Erik F Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics. 2003, pp. 142–147.

TRITA -EECS-EX-2018:711