國立政治大學社群網路與人智計算
國際研究生博士學位學程

博士學位論文

遞歸及自注意力類神經網路之強健性分析

Analysis of the Robustness of Recurrent and Self-attentive Neural Networks

博士班學生：謝育倫　撰
指導教授：許聞廉　博士
劉昭麟　博士

中華民國 109 年 6 月

DOI:10.6814/NCCU202001426

致謝

本論文能夠完成，需要感謝非常多一路上幫助我的師長、同儕、以及家人。首先要謝謝我的指導教授—許聞廉博士及劉昭麟博士，他們總是給我空間來探索各種可能的研究方向，也願意提供充沛的資源支援。如果沒有老師的幫忙，就不會有今天的成果。同時，感謝加州大學謝卓叡教授，在我前往美國擔任訪問學者期間的討論與指導，讓我們的研究結晶能夠發表在國際頂尖會議中。而博士班的研究生涯，則起源於進入中研院資訊所智慧型代理人系統實驗室。在這近十年的經歷中，有幸能夠接受許多博士前輩的帶領，包含李政緯、施政緯、姜天戩、劉士弘、張詠淳等，不僅在學術領域上給予我極大的幫助，也傳授我許多軟體技術的要點。並且要感謝實驗室其他同仁：翠玲、秋珍、照玲、永瑜、庭豪、千嘉、柏廷、堃育、欣裕、乃文、國豪、明祥、俊翰、書豪、友珊、新蓉、瑩棻、佳蓉、瑞祥、俊宏、茂彰、建勤，以及更多共事過的伙伴。是大家的合作無間，造就了我們這個研究團隊今天的成果。另外，我也非常感激二十餘年來的同學及好友—大成，除了在求學階段有過許多向他討教的機會，更是在我赴美訪問時大力相助，我才能夠平安快樂的度過那段時光，這段恩澤將永銘於我心。最後，要感謝雨芬，在我們相知相惜的這段不算長的日子，願意給我鼓勵和信任，還有不辭辛勞的奔波，多虧有了這些支持，我才能在有限的時間

DOI:10.6814/NCCU202001426

內完成學業。最重要的，是感謝我的父親、母親，自我來到這個世界時，就持續無條件的照顧和付出，讓我能無憂無慮的學習，這麼深厚的恩情，實在無以回報，謹以此論文獻給他們。

DOI:10.6814/NCCU202001426

中文摘要

本文主要在驗證目前被廣泛應用的深度學習方法，即利用類神經網路所建構的機器學習模型，在自然語言處理領域中之成效。同時，我們對各式模型進行了一系列的強健性分析，其中主要包含了觀察這些模型對於對抗性（adversarial）輸入擾動之抵抗力。更進一步來說，本文所進行的實驗對象，包含了近期受到許多注目的 Transformer 模型，也就是建構在自我注意力機制之上的一種類神經網路，以及目前常用的，基於長短期記憶 (LSTM) 細胞所搭建的遞歸類神經網路等等不同網路架構，觀察其應用於自然語言處理上的結果與差異。在實驗內容上，我們囊括了許多在自然語言處理領域中最常見的工作，例如：文本分類、斷詞及詞類標註、情緒分類、蘊含分析、文件摘要及機器翻譯等。結果發現，基於自我注意力的 Transformer 架構在絕大多數的工作上都有較為優異的表現。除了使用不同網路架構並對其成效進行評估，我們也對輸入之資料加以對抗性擾動，以測試不同模型在可靠度上的差異。另外，我們同時提出一些創新的方法來產生有效的對抗性輸入擾動。更重要的是，我們基於前述實驗結果提出理論上的分析與解釋，以探討不同類神經網路架構之間強健性差異的可能來源。

關鍵字：自我注意力機制、對抗性輸入、遞歸類神經網路、長短期記憶、強健性分析

DOI:10.6814/NCCU202001426

Abstract

In this work, we focus on investigating the effectiveness of current deep learning methods, also known as neural network-based models, in the field of natural language processing. Additionally, we conduct robustness analysis of various neural model architectures. We evaluate the neural network's resistance to adversarial input perturbations, which in essence is replacing the input words so that the model might produce incorrect results or predictions. We compare the differences between various network architectures, including the Transformer network based on the self-attention mechanism, and the commonly employed recurrent neural networks using long short-term memory cells (LSTM). We conduct extensive experiments that include the most common tasks in the field of natural language processing: sentence classification, word segmentation and part-of-speech tagging, sentiment classification, entailment analysis, abstractive document summarization, and machine translation. In the process, we evaluate their effectiveness as compared with other state-of-the-art approaches. We then estimate the robustness of different models against adversarial examples through five attack methods. Most importantly, we propose a series of innovative methods to generate adversarial input perturbations, and devise theoretical analysis from our

DOI:10.6814/NCCU202001426

observations. Finally, we attempt to interpret the differences in robustness between neural network models.

Keywords: Robustness, Adversarial Input, RNN, LSTM, Self Attention

DOI:10.6814/NCCU202001426

Contents

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

List of Tables

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

List of Figures

DOI:10.6814/NCCU202001426

DOI:10.6814/NCCU202001426

Chapter 1

Introduction

1.1 Motivation

Research on Artificial Intelligence (AI) has been increasingly influential in recent years. In particular, Natural Language Processing (NLP), a field that combines linguistic theories and techniques in computer science, is poised to become the center of all AI applications. It is not surprising, since the ability to utilize language is a crucial part of the human life as well as intelligence. Without NLP, integrating machines with the human intelligence would be very infeasible. Meanwhile, machine learning (ML) methods have been widely applied to a broad spectrum of problems in this field.

Currently, prominent ML models rely on neural network-based architectures to obtain state-of-the-art outcomes on many NLP jobs, *e.g.*, classification of documents, sentiments, and machine translation (MT). Notably, self-attention-based models are receiving a surge of recognition in the past few years. This type of models, including Transformer [70] as well as "Bidirectional Encoder Representations from Transformers," or *BERT* [18], rely on the attention mechanism [46] to learn a context-dependent word representation. In particular, BERT is

DOI:10.6814/NCCU202001426

proposed more recently in an attempt to encode even richer contextual information into the vector representation of words.

Compared with recurrent neural networks (RNN), these Transformer-based models have a more efficient encoding speed while maintaining the capacity of incorporating a broader contextual information. A common pre-training method of this type of models involves a uni-directional language model objective, whereas BERT exploits a new process that randomly drops some of the input words and an alternative objective that tries to determine the missing ones using only its neighboring tokens. Particularly, BERT is pre-trained in the way that utilizes multiple goals to force its encoding capability. This novel design prompts the model to learn a combined representation that fuses both the left and right context, creating a bidirectional feature extractor. In addition, the alternative "next sentence prediction" objective is also included, where the model has to classify whether a pair of input sentences is extracted from two consecutive locations in the training corpus. Subsequently, this pre-trained model can be easily tuned to perform a wide variety of down-stream tasks. The BERT model obtains state-of-the-art results on numerous NLP problems, *e.g.*, classification and question answering. Oftentimes, we see it surpassing task-specific models that are carefully engineered. Thus, it is fast becoming a core element in solving a wide variety of NLP tasks.

Nevertheless, the structure, *i.e.*, self-attention, underlying BERT and Transformer requires a further investigation. Given their success, the robustness of these model against adversarial attacks compared with other neural networks is yet to be studied. An adversarial attack involves the application of tiny perturbations on the input of the model, thereby creating a so-called adversarial example where the change is humanly imperceptible. This example can trick the model into making an error in prediction [24]. Unlike those in computer vision research, it is challenging to come up with a textual adversarial example that is not easily detected by humans

DOI:10.6814/NCCU202001426

and misguides the machine [3, 39, 54, 82]. However, some recent work unveil that these models are vulnerable to adversarial examples with acceptable quality [5, 37]. The adversarial inputs can be precisely recognized by human evaluation, and at the same time trick ML models to produce incorrect results. For example, a review that says "We had a great experience 6 months ago, but last night was strikingly different." It is imaginable that this review is implying that the author did not have a positive sentiment. However, a machine can be easily fooled by the majority of positive-sounding words in the sentence, and therefore determine that this is a positive sentence. The fragility of statistical machine learning models resembles even the traditional rule-based ones. They can be unaware of the meaning of novel data that was not present in the training set. In other words, the generality of these models is not guaranteed. These problems urge us to investigate the robustness of current deep learning models when applied to NLP applications.

1.2 Research Objectives

The main focus of the current dissertation pertains to evaluating the capabilities of deep neural networks when used for a number of fundamental tasks in natural language processing. Particularly, we investigate neural networks with different structures, including recurrent and self-attentional models. The goal of this work is manifold. Most importantly, we attempt to answer the following research question:

1. When compared with traditional recurrent networks, are self-attentive (Transformer) models more robust to adversarial examples?

2. Why does one network structure outperform the other with regards to robustness?

DOI:10.6814/NCCU202001426

3. Can we use the attention weights in self-attentive neural networks to exploit vulnerability in these models?

In short, this work verifies the robustness of recurrent and self-attentive models. This is accomplished through performing adversarial attacks and analyzing their effects on the model prediction. Besides, we evaluate the possibility of employing these context-dependent word representations to devise metrics for measuring the semantic similarity between adversarial and actual input sentences. The experiments in this dissertation include two common self-attentive neural networks, (a) Transformer for neural machine translation, and (b) BERT for sentiment and entailment classification. The compared methods are mainly recurrent neural networks.

This work is unique in a number of aspects. First, we examine the robustness of uni- and bi-directional self-attentive model as compared to RNNs. We provide detailed observations of the internal variations of models under attack. And, we devise novel attack methods that take advantage of the embedding distance to maximize semantic similarity between real and adversarial examples.

To the best of our knowledge, this work brings forth the following contributions.

1. We conduct comprehensive experiments to examine the robustness of LSTM, Transformer, and BERT. Our results show that both self-attentive models, whether pre-trained or not, are more robust than LSTM models.

2. We propose novel algorithms to generate more natural adversarial examples that both preserve the semantics and mislead the classifiers.

3. We provide theoretical explanations to support the statement that self-attentive structures are more robust to small adversarial perturbations.

DOI:10.6814/NCCU202001426

1.3 Outline

The remaining chapters are organized as follows. Chapter 2 begins by mentioning a spectrum of essential knowledge which help the reader to grasp the main concepts of this dissertation. They include prior work on machine learning, especially regarding neural networks, methods and tasks that are prominent in natural language processing. Next, Chapter 3 contains descriptions of our approaches for adapting pre-trained models on various NLP-related tasks, including classification, sequence labeling, sentiment analysis, entailment, and machine translation. Subsequently, the experiments on these tasks are presented in Chapter 4. We provide some discussions on the theoretical aspect of the experiments on the robustness of self-attentive models as compared with recurrent neural networks in Chapter 5. Finally, we conclude this work with Chapter 6, in which we summarize the results from previous chapters as well as propose advances that can be made in the future.

1.4 Publications

The current dissertation is based upon many previous work by the author and other collaborators. They are listed below.

1. **Yu-Lun Hsieh**, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, Cho-Jui Hsieh, "On the Robustness of Self-Attentive Models,", in Proceedings of *the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*.

2. **Yu-Lun Hsieh**, Yung-Chun Chang, Nai-Wen Chang, Wen-Lian Hsu, "Identifying Protein-protein Interactions in Biomedical Literature using Recurrent Neural Networks with Long Short-Term Memory," in *Proceedings of the Eighth International Joint Conference on*

DOI:10.6814/NCCU202001426

Natural Language Processing (IJCNLP 2017).

3. **Yu-Lun Hsieh**, Yung-Chun Chang, Yi-Jie Huang, Shu-Hao Yeh, Chun-Hung Chen, Wen-Lian Hsu, "MONPA: Multi-objective Named-entity and Part-of-speech Annotator for Chinese using Recurrent Neural Network," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

4. **Yu-Lun Hsieh**, Shih-Hung Liu, Kuan-Yu Chen, Hsin-Min Wang, Wen-Lian Hsu, Berlin Chen, "Exploiting Sequence-to-Sequence Generation Framework for Automatic Abstractive Summarization," in *Proceedings of the 28th International Conference on Computational Linguistics and Speech Processing (ROCLING 2016)*.

Other work that are related to this topic includes:

5. **Yu-Lun Hsieh**, Yung-Chun Chang, Chun-Han Chu, Wen-Lian Hsu, "How Do I Look? Publicity Mining From Distributed Keyword Representation of Socially Infused News Articles", in *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media (collocated with EMNLP 2016)*.

6. **Yu-Lun Hsieh**, Shih-Hung Liu, Yung-Chun Chang, Wen-Lian Hsu, "Neural Network-Based Vector Representation of Documents for Reader-Emotion Categorization," in *Proceedings of the 2015 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 569–573, San Francisco, CA, USA, 2015.

7. **Yu-Lun Hsieh**, Shih-Hung Liu, Yung-Chun Chang, Wen-Lian Hsu, "Distributed Keyword Vector Representation for Document Categorization," in *Proceedings of the 2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI),* pp. 245–251, 2015.

DOI:10.6814/NCCU202001426

The following publications are also completed during the course of the Ph.D.

8. Zheng-Wen Lin, Yung-Chun Chang, Chen-Ann Wang, **Yu-Lun Hsieh**, Wen-Lian Hsu "CIAL at IJCNLP-2017 Task 2: An Ensemble Valence-Arousal Analysis System for Chinese Words and Phrases," in *Proceedings of the IJCNLP 2017, Shared Tasks*.

9. Shih-Hung Liu, Kuan-Yu Chen, **Yu-Lun Hsieh**, Berlin Chen, Hsin-Min Wang, Hsu-Chun Yen, Wen-Lian Hsu, "Exploiting graph regularized nonnegative matrix factorization for extractive speech summarization," in *Proceedings of APSIPA 2016*.

10. Shih-Hung Liu, Kuan-Yu Chen, **Yu-Lun Hsieh**, Berlin Chen, Hsin-Min Wang, Hsu-Chun Yen, Wen-Lian Hsu, "Exploring Word Mover' s Distance and Semantic-Aware Embedding Techniques for Extractive Broadcast News Summarization." in *Proceedings of INTERSPEECH 2016*.

11. Ting-Hao Yang, **Yu-Lun Hsieh**, You-Shan Chung, Cheng-Wei Shih, Shih-Hung Liu, Yung-Chun Chang, Wen-Lian Hsu, "Principle-Based Approach for Semi-Automatic Construction of a Restaurant Question Answering System from Limited Datasets," in *Proceedings of the 2016 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 520–524, Pittsburgh, PA, 2016.

12. Nai-Wen Chang, Hong-Jie Dai, **Yu-Lun Hsieh**, Wen-Lian Hsu, "Statistical Principle-Based Approach for Detecting miRNA-Target Gene Interaction Articles," in *Proceedings of the 2016 IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE)*, 2016.

DOI:10.6814/NCCU202001426

Chapter 2

Background and Related Work

In this chapter, we briefly review the basis of natural language processing research, which is the main target of this research. Then, we describe fundamental technologies that is later utilized in the rest of this work. We also introduce the readers to more details of the essential model, *i.e.*, neural networks, that is under investigation of this dissertation.

2.1 Natural Language Processing

Natural language processing (NLP) has been an essential part of the development of artificial intelligence since as early as the 1950s [69]. The main aim of this field is to design models and methods for a computer, or any machinery, to store, process, and eventually understand human languages.

There are many levels of processing when dealing with language. Depending on the language, one may need to perform lemmatization or stemming first. These steps break up words into smaller, meaningful parts. Another related work is called "word segmentation," where one has to find word boundaries within a sentence in which they are not obvious. Languages

DOI:10.6814/NCCU202001426

like Chinese, Japanese, and Thai are some of the examples of this category. Part-of-speech (POS) labeling, or tagging, refers to assigning a label that represents the part-of-speech for each word. POS are classes of a word for the purpose of grammatical description. Some of those classes include: the verb, the noun, the pronoun, the adjective, the adverb, the preposition, the conjunction, the article, and the interjection [29]. Other major components of NLP include:

Syntactic (constituency) parsing involves creating a structured representation of the syntactic relationships of the words.

Dependency parsing aims at identifying the subject, object, and predicates of a sentence. It is done by labeling the relationship between one word and another.

Named entity recognition mainly focuses on finding the entities in a sentence, including persons, places, organizations, etc.

Sentiment analysis or opinion mining, refers to identify the affective content in text. It is commonly employed to analyze product reviews, survey responses, social media, etc., for use in applications such as marketing or customer service.

Entailment detection targets to find out the directional relationship between statements. We define a piece of text T and a hypothesis H. If T entails H, it is understood as if one reads T, one would infer that H is very likely to be true. The directionality factor means that the reverse does not hold, *i.e.*, H does not necessarily entail T.

Machine translation model generates the translation of one language to another based on the training data in a bilingual corpus. It can be traced back to the idea proposed by Weaver [72]

DOI:10.6814/NCCU202001426

that a machine can be utilized to handle this task. Traditionally, statistical machine translation was the common approach. In recent years, the application of neural networks has boosted the performance to a new peak.

Summarization In the past, more attention has been paid to **extractive** summarization, while **abstractive** summarization are rather rare. In view of the recent success of deep learning, the research on abstractive summarization has been growing. Recent literature has preliminarily verified the effectiveness of RNN on rewritten summarization of documents. Moreover, the contribution of the attention mechanism is also noticed by many. The characteristic of this mechanism is that it can increase the weight of key segments while generating text, thereby composing a better summary.

2.2 Neural Networks

The NLP community is widely adopting Artificial Neural Networks (ANN) for various topics in this field. Thus, we begin this section by supplying an overview of neural networks and its elementary ingredients.

An ANN can be regarded as a series of functions that are strung together, in which the majority are non-linear. It is important to note that, an ANN can learn to replicate linear or logistic regression, as well as other fundamental statistical machine learning models [40]. To illustrate, we consider the logistic regression problem an example here. A multi-class logistic regression can be represented as:

$$\begin{aligned} f(\mathbf{x}) &= \boldsymbol{W}\mathbf{x} + \mathbf{b} \\ g(\mathbf{y}) &= \mathrm{softmax}(\mathbf{y}) \end{aligned} \tag{2.2.1}$$

DOI:10.6814/NCCU202001426

where $\boldsymbol{W} \in \mathbb{R}^{C \times d}$ denotes the weights in an ANN in matrix form, $\mathbf{x} \in \mathbb{R}^d$ the input vector, $\mathbf{b} \in \mathbb{R}^C$ the bias, and $\mathbf{y} \in \mathbb{R}^C$. C is the number of classes and d is the dimensionality of the input. We subsequently use θ as a shorthand to denote $\{\boldsymbol{W}, \mathbf{b}\}$, the set of parameters of this ANN. As such, $g(f(\mathbf{x}))$ represents the logistic regression in the form of a composition of f and g. When we use ANN as an approach to this, the f is modeled by a fully connected NN, and g an activation function, in this case, $\mathrm{softmax}$. Note that the activation function of ANNs is non-linear. Technically, an ANN contains more than one "layer" of the above computation, therefore called "deep neural networks," or DNN. These layers are connected or stacked together, with the output of one of them being the input of the next. The "hidden" layers, or the ones that are between the input and output, typically use activation functions other than the $\mathrm{softmax}$, *e.g.*, $\mathrm{ReLU}, \tanh$. For the output layer, the $\mathrm{softmax}$ and $\mathrm{sigmoid}$ functions are commonly used, due to the assumption that the output layer of an ANN can be considered as categorical or Bernoulli distribution. Thus, the linear as well as logistic regression can be approximated by an ANN with just one layer, in which the former uses the identity function and the latter non-linear activation function. Note that fully connected feed-forward ANN can sometimes be referred to as a multilayer perceptron (MLP) [64].

$$\begin{aligned} \mathbf{h} &= \sigma(\boldsymbol{W}_1 \mathbf{x} + \mathbf{b}_1) \\ \mathbf{y} &= \mathrm{softmax}(\boldsymbol{W}_2 \mathbf{h} + \mathbf{b}_2), \end{aligned} \tag{2.2.2}$$

where the σ denotes the activation function. It is worthy of mentioning that the layers can typically have independent weight matrices. Here, the first hidden layer contains a weight matrix $\boldsymbol{W}_1$ and bias $\mathbf{b}_1$. The process of obtaining $\mathbf{h}$, or the output of the layer, and feeding to the following layer as its input is called the "forward propagation". As a result, the output of the

DOI:10.6814/NCCU202001426

final layer, *i.e.*, $\mathbf{y}$, can be thought of as the output of the whole ANN. Recall that, mathematically the composition of more than one linear functions is equivalent to another linear function. It is perceivable that the model only has limited expressive power if we build it in this manner. Thus, the use of non-linear activation functions is at the heart of the success of deep neural networks.

However, some more advanced ANNs can be designed to share those weights, sometimes referred to as "weight tying." It is often used as a way of reducing the number of parameters in a model, as well as having the benefit of creating an inductive bias. Such bias may increase the ability of the model to generalize, and has been examined in previous work [62]. For example, a general pre-trained model can be transferred to various down-stream tasks due to the generality. This technique is widely used in current deep learning models.

2.2.1 Activation Functions

Activation functions are mathematical equations that determine the output of a cell in a neural network. Much like a real neural network in organisms, this function is imposed upon each neuron in the network so that the output value represents the activated state. In addition, this function can sometimes act as a regularization of the output.

For current ANNs, we require an extra characteristic of the activation function, which is that it must be differentiable. One of the most common functions is the sigmoid, denoted by the following:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2.3}$$

Another function, the softmax, is one that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the

DOI:10.6814/NCCU202001426

exponentials of the input. Specifically,

$$\text{softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \text{ for } i = 1, \ldots, K \text{ and } x = (x_1, \ldots, x_K) \in \mathbb{R}^K \tag{2.2.4}$$

The sigmoid and softmax are generally adopted at the last, *i.e.*, output layer, of the ANN. As for the hidden layers in between, we mostly employ the rectified linear unit (ReLU) function. It is written as the following:

$$\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x}) \tag{2.2.5}$$

Yet another activation function, the hyperbolic tangent or $\tanh$, is clearly defined and requires no further explanation. It outputs values in the range $(-1, 1)$.

Recently, other functions such as the Exponential Linear Units (ELU) and the Gaussian Error Linear Unit (GELU) [19, 30] has been proposed. ELU is aimed at speeding up the learning process of deep neural networks and at the same time retaining a high classification accuracy. Part of the ELU is similar to ReLU, where the identity function is used to handle the positive section of the input values, in order to tackle the vanishing gradient problem. On the other hand, ELU has the unique property of allowing negative values. This trait serves as a normalization factor very similar to the batch normalization process, which shifts the average activation of the units towards zero. But, unlike batch normalization, it requires no extra computation overhead.

2.2.2 Recurrent Neural Networks

Among different types of ANNs, the recurrent neural network (RNN) is one that is suitable for learning sequential input [21]. Recurrent neural networks have recently become a popular solution for single sequence as well as sequence-to-sequence tasks. In particular, prominent

DOI:10.6814/NCCU202001426

network structures such as the 'seq2seq' proposed by [4, 15, 65] are increasingly being applied to a wide variety of problems. Moreover, some tasks that were considered as difficult are seeing explosive advances, including machine translation (MT) and language modeling (LM), when deep neural networks are incorporated. [38, 46] Typically, an encoder-decoder scheme is adopted to deal with these tasks, where the input sequence is encoded by an encoder, and the subsequent decoder generates a (sequential) output.

Alternatively, we can view RNNs as a feed-forward neural network in which all layers share the same set of parameters. But, note that, rather than a fixed number of layers, the 'depth' (or number of layers) of this type of NN is dependent on the length of the input sequence. We can see that each element in the input sequence can be treated as the input of each layer. To be more formal, we can define an RNN as maintaining a vector $\boldsymbol{W}_h\mathbf{x}_t$, which is a hidden state or memory to store at each time step t. Then, upon receiving input at time step t, the network updates its state as the following:

$$\begin{aligned} \mathbf{h}_t &= \sigma_h(\boldsymbol{W}_h\mathbf{x}_t + \boldsymbol{U}_h\mathbf{h}_{t1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \sigma_y(\boldsymbol{W}_h\mathbf{x}_t + \mathbf{b}_y) \end{aligned} \tag{2.2.6}$$

where σ_h and σ_y are activation functions. We can see from the above formulation that the weights $\boldsymbol{U}_h$ is to transform the previous hidden state $\mathbf{h}_{t1}$, and $\boldsymbol{W}_h$ the current input $\mathbf{x}_t$. A bias term $\mathbf{b}_h$ can also be added. These calculations update the state vector $\mathbf{h}_t$. Subsequently, the RNN produces an output $\mathbf{y}_t$. However, an RNN can be less effective when modeling a sequential input if the number of time steps exceeds a certain amount, as we can induce from the above formulation.

Nevertheless, an RNN depends on the previous calculation results to produce the next one. So, an increasing amount of efforts have been devoted to find a mechanism that can replace recurrence. One possible direction of research is to use attention [4]. The motivation behind this approach is that it can combine the efficiency of attention computation with the ability to

DOI:10.6814/NCCU202001426

learn positional information. It has been shown [70] to achieve outstanding performances on a multitude of language pairs in MT. We will introduce attention-based models later in this chapter.

2.2.3 Long Short-term Memory

A neural network cell named Long Short-term Memory, or LSTM, is proposed in an attempt to solve the deficiency of simple RNNs in learning a sequence of a longer length [27, 31]. It is evidences by the experimental results that LSTM can remember a longer span of the sequential input, as compared to traditional RNNs. This trait is important especially for NLP applications.

In essence, an LSTM is an augmented RNN with extra weights to determine the amount of information to "remember" as well as "forget." They are done through the implementation of a forget gate $\mathbf{f}_t$, an input gate $\mathbf{i}_t$, and an output gate $\mathbf{o}_t$. The values of all these gates are dependent on the input $\mathbf{x}_t$ and previous memory $\mathbf{h}_t$. In this way, the cell learns to determine the portion of its state to keep or discard through the forget gate. Formally, we recall the current input $\mathbf{x}_t$, the previous output $\mathbf{h}_{t-1}$, and the current cell state $\mathbf{c}_t$ as defined in the previous paragraph. Using the following formulas, we enable the model to learn what to forget in the past and remember in the moment.

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i \mathbf{x}_t + \boldsymbol{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\boldsymbol{W}_f \mathbf{x}_t + \boldsymbol{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o \mathbf{x}_t + \boldsymbol{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(W_c \mathbf{x}_t + \boldsymbol{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned}
\tag{2.2.7}
$$

where σ designates the sigmoid function and "$\circ$" means the element-wise product between

DOI:10.6814/NCCU202001426

vectors.

Graves et al. [26] proposed an intuitive extension to the LSTM, called the Bidirectional LSTM, or *BiLSTM*. In essence, it involves creating two separate LSTMs, of which one receives the original input sequence and the other a reversed sequence. These two LSTMs learn to model the sequential input separately and independently. This model is later widely used in virtually all NLP models [79].

2.2.4 Training

The training (learning) phase of a neural network currently relies on stochastic gradient descent. However, a typical network consists of more than one layer, preventing us from calculating the gradient of the loss function. Thus, a dynamic process named "back propagation," or "backprop (BP)". is commonly adopted [63].

BP adopts the chain rule of calculus to determine the gradient of vectors. Let $\boldsymbol{x} \in \mathbb{R}^m$ be the input to a neural network, $\boldsymbol{y} \in \mathbb{R}^n$ be the output of the penultimate layer, we can formulate the network as:

$$\begin{aligned} \boldsymbol{y} &= g(\boldsymbol{x}) \\ z &= f(\boldsymbol{y}) = f(g(\boldsymbol{x})) \end{aligned} \tag{2.2.8}$$

where z is a scalar output of the network. The gradient of z with respect to every element x_i in $\boldsymbol{x}$ can be written as:

$$\frac{\partial z}{\partial x_i} = \Sigma_j \frac{\partial z}{\partial y_j} \times \frac{\partial y_j}{\partial x_i} \tag{2.2.9}$$

Since we are using vectors as input, the gradients $\nabla_{\boldsymbol{x}} z$ can be computed by the following multiplication:

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z \tag{2.2.10}$$

DOI:10.6814/NCCU202001426

where the Jacobian matrix of g is denoted by $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$, and $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \in \mathbb{R}^{n \times m}$. This matrix contains all partial derivatives. As described in [25], for all operations in the forward pass of the NN, the BP derives the Jacobian-gradient product. Let function

$$J = \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \tag{2.2.11}$$

be the loss function of a certain task performed by a neural network that we need to minimize. This NN has K layers with weights $\boldsymbol{W}_k$ and biases $\boldsymbol{b}_k$, where $k \in \{1, \cdots, K\}$. First we perform forward calculation using the input $\boldsymbol{x}$ starting from the first layer and ending at the last one (output), yielding the output vector $\hat{\boldsymbol{y}}$. Then, we obtain the loss by J. BP is therefore acting in a reversed order. It starts by calculating the gradients $\nabla_{\hat{\boldsymbol{y}}} J$ with respect to the output $\hat{\boldsymbol{y}}$. Subsequently for all previous layers, it obtains the partial derivatives of parameters $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ for each layer until the first one is reached. It is perceivable that, in such a process, the derivatives of the deeper layers (close to the output) must be obtained first before the shallower layers can be considered. This is due to the fact that the values of deep layers depend on those from shallow layers. Finally, the SGD algorithm applies the gradients onto the parameters and completes the optimizing step. Typically, this procedure is repeated for a certain amount of "epochs," or traversals of the entire training dataset.

Note that, when training RNNs, the gradients should be passed along the time step axis and not through the depth-wise procedure of other types of NNs. To do that, we must employ a technique known as back-propagation through time (BPTT) [73]. But the practical limitation of hardware prevent us from doing BP indefinitely. So, we normally set a certain window on the time axis for BP to operate in. Interestingly, one might think a wider window can help the RNN to see and model a wider context. Whereas in practice, we often find that the network is unable

DOI:10.6814/NCCU202001426

to learn anything. After careful inspection, it is found that the problem of *exploding* or *vanishing gradients* exists in these situations. Consider for a moment the forward operation of an RNN, in which the state vector is continuously being updated by multiplication of the weights. Thus, when BP is in action, the gradients would also undergo the same process multiple times. It is imaginable that these gradients may become exceedingly large (explode) or small (vanish). As a result, the network is unable to be optimized. Therefore, the LSTM (mentioned in Section 2.2.3) is proposed to alleviate the exploding or vanishing gradient problem.

2.3 Attention Mechanisms

The "attention" in neural networks can be thought of as a type of weighting. There are multiple functions to obtain attention scores, but the additive method [4] and multiplicative method [46] are among the most widely used ones. The multiplicative method is sometimes factored by $\frac{1}{\sqrt{d_k}}$, which is used in BERT-related models. In particular, the form that is commonly used is called "Scaled Dot-Product Attention". Consider that, in an attention block, the input is a series of vectors named "queries" and "keys" with dimension d_k, and "values" of dimension d_v. Then, the dot product between a query and all keys are calculated and normalized by dividing with $\sqrt{d_k}$. Finally, the softmax function is applied and the weights on the values can be output. Note that, we can parallelize the attention calculation by computing them on a batch of inputs at the same time. More specifically, the query, key, and value vectors are collated into matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, respectively. We then perform matrix operations to obtain outputs as:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{2.3.1}$$

DOI:10.6814/NCCU202001426

Conceptually, attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

On the other hand, additive attention employs a fully connected NN to determine the compatibility or similarity between two vectors. The score of additive attention is calculated as follows:

$$\text{Attention}_{add}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{v}_a^\top \tanh(\boldsymbol{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i]) \tag{2.3.2}$$

where $\boldsymbol{s}, \boldsymbol{h}$ are state vectors and $\boldsymbol{v}, \boldsymbol{W}$ are parameters that the model needs to learn. We can observe the above definitions and find that multiplication in dot-product attention can take advantage of modern GPU hardware to perform fast calculations. This is one of the main reasons for recent improvements in using attention for language modeling.

2.3.1 Self-Attention

Recently, Vaswani et al. [70] proposed the "Transformer" model, which is a novel method that solely depends on self-attention to learn vector representations of a sequence. The heart of Transformer is a unit of multi-head self-attention mechanism. It *transforms* the input vectors into a representation formed by multiple mixtures learned by the model. For each *head*, The input is first linearly projected by a set of three weight matrices as in the previous section to three vectors, (Q, K, V). Then, an attention weight is calculated using the dot-product attention in Eq. 2.3.1. The reason it is called the "self"-attention is that the attended elements are the input sequence itself. Moreover, note that the number of *head* effectively indicates the number of weight matrix sets. Figure 2.1 shows a schematic of the Transformer model. The trait of

DOI:10.6814/NCCU202001426

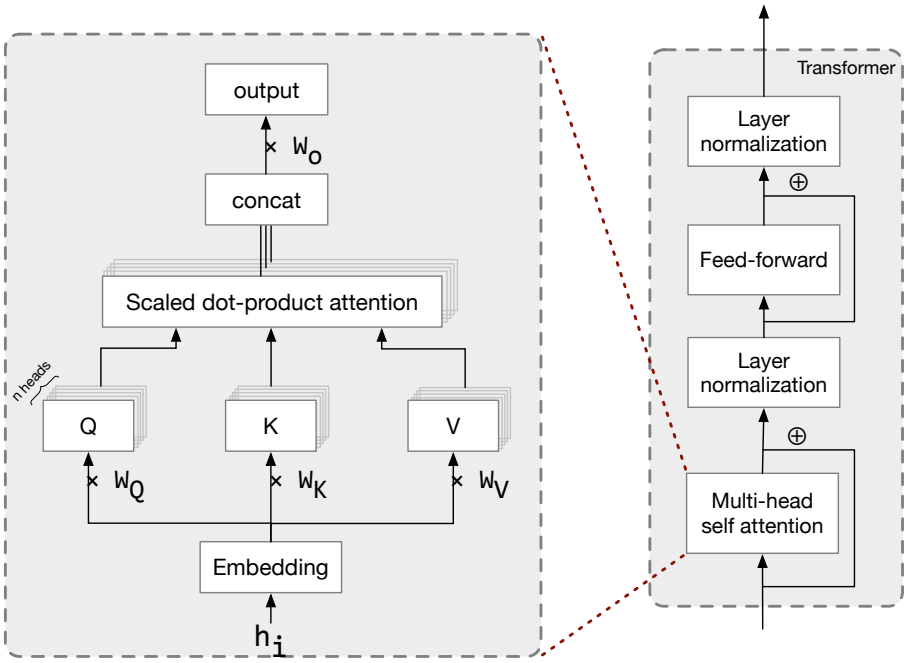


Figure 2.1: The multi-head attention module in a Transformer block.

this type of models is that it frees us from the recurrent part of previous neural networks or even convolution calculations. The Transformer model utilize entirely the attention weights to denote global correspondence between input and output symbols (words). Thus, the degree of parallelization can be much higher than RNNs. The performance is indicated by training a machine translation system that achieves state-of-the-art outcomes in just 12 hours.

However, this type of model does not come without weaknesses. First, it cannot directly learn the order of the input due to the fact that there is no recurrent state or convolution. So, another new type of embedding, *Position Embedding*, is incorporated to represent the relative order of the elements in the input sequence. They are defined as follows.

$$\mathbf{x}_i = (\mathrm{emb}_{word} i \oplus \mathrm{emb}_{tag} i) + \mathrm{emb}_{pos} i \tag{2.3.3}$$

where $\mathrm{emb}_{pos} i$ is called the *Position Embedding* of the i-th position. We use the sine and cosine

DOI:10.6814/NCCU202001426

functions of different frequencies to compose position embeddings as stated in [70]. It is added to the word embeddings which represent linguistic information of the words, with dimensions identical to other embeddings so that they are compatible.

Specifically, these functions are used to encode the position information [70]:

$$\begin{aligned} \mathrm{emb}_{\mathrm{pos}2i} &= sin(pos/10000^{2i/d_{\mathrm{model}}}) \\ \mathrm{emb}_{\mathrm{pos}2i+1} &= cos(pos/10000^{2i/d_{\mathrm{model}}}) \end{aligned} \tag{2.3.4}$$

where pos is the position and i is the dimension. We can interpret this formulation as using a sinusoid to encode one dimension with position embedding. The wavelengths denoted by these functions constitute a geometric sequence (a sequence of numbers where each one is determined by multiplying the previous one by a fixed, non-zero value) from 2π to $(10000 \cdot 2\pi)$. It is hypothesized that this formulation can incorporate the relative position of the input elements into the embeddings. The reason is explained as follows. Suppose we have a word embedding $\mathrm{emb}_{\mathrm{pos}}$, the embedding of a word that is k steps away can be denoted by a linear combination of $\mathrm{emb}_{\mathrm{pos}+k}$.

It is worthy of noting that there are various means of learning positional information [23]. The learned positional embeddings [23] are examined and the results showed that they have virtually no difference [70]. The sine and cosine functions are eventually selected due to their potential in modeling the indefinitely long sequence that may not be present in the training data. The characteristics of these functions can help the network extrapolate to unseen lengths.

DOI:10.6814/NCCU202001426

2.4 Adversarial Attack

Robustness of neural network models has been a prominent research topic since Szegedy et al. [66] discovered that CNN-based image classification models are vulnerable to adversarial examples. An abundant amount of work has been dedicated to investigating the robustness of CNN models against adversarial attacks [7, 12, 24, 54]. However, attempts to examine the robustness of NLP models are relatively few and far between. Previous work on attacking neural NLP models include using Fast Gradient Sign Method [24] to perturb the embedding of RNN-based classifiers [43, 53], but they have difficulties mapping from continuous embedding space to discrete input space. Ebrahimi et al. [20] propose the 'HotFilp' method that replaces the word or character with the largest difference in the Jacobian matrix. Li et al. [41] employ reinforcement learning to find the optimal words to delete in order to fool the classifier. More recently, Yang et al. [77] propose a greedy method to construct adversarial examples by solving a discrete optimization problem. They show superior performance than previous work in terms of attack success rate, but the greedy edits usually degrade the readability or significantly change the semantics. Alzantot et al. [3] propose to use a pre-compiled list of semantically similar words to alleviate this issue, but leads to lower successful rate as shown in our experiments. We thus include the latest greedy and list-based approaches in our comparisons.

In addition, the concept of adversarial attacks has also been explored in more complex NLP tasks. For example, Jia and Liang [37] attempt to craft adversarial input to a question answering system by inserting irrelevant sentences at the end of a paragraph. Cheng et al. [14] develop an algorithm for attacking seq2seq models with specific constraints on the content of the adversarial examples. Belinkov and Bisk [5] compare typos and artificial noise as adversarial input to machine translation models. Also, Iyyer et al. [35] propose a paraphrase generator

DOI:10.6814/NCCU202001426

model to generate legitimate paraphrases of a sentence. However, the semantic similarity is not guaranteed. In terms of comparisons between LSTM and Transformers, Tang et al. [67] show that multi-headed attention is a critical factor in Transformer when learning long distance linguistic relations.

2.4.1 Pre-training and Multi-task Learning

The widely adopted workflow for BERT-related models involves two steps, *i.e.*, pre-training (self-supervised) and fine-tuning (supervised). The first pre-training procedure involves two targets, *i.e.*, masked language modeling (MLM) and next sentence prediction (NSP). MLM regards the prediction of randomly masked input words, and NSP aims at predicting the relationship of two input sentences, namely, does the latter follow the former in the original corpus. Subsequently, the model is fine-tuned for different downstream tasks, where fully-connected networks are added after the final encoding layer per the requirements of various end tasks.

Such a scheme can be regarded as a type of multi-task learning. In recent ML research, this approach has been proven successful for a wide spectrum of applications that go beyond NLP [16]. Traditionally, ML model focuses on a single task in the training process. In doing so, we are stripping it of information potentially useful for other tasks as well. If the supervised goal of another task contains knowledge which can assist the model to learn quicker and better, it is helpful to train both tasks together. Therefore, recent approaches attempt to construct a common, general representation scheme for all tasks before specifying the supervised goals. As the experiments showed, MTL can boost the model's ability to generalize across different tasks [8]. More specifically for NLP tasks, pre-training the word representation models such as ELMo or GPT-2 [60, 61] have been verified to greatly boost their effectiveness in a broad variety

DOI:10.6814/NCCU202001426

of applications. Note that, the pre-training loss of these models may be different. But, as long as it is designed to incorporate helpful auxiliary information such as linguistic knowledge, the resulting model can be stronger when learning a new task.

2.5 Evaluation Metrics

Throughout this dissertation, we use standard metrics in NLP to evaluate the performance of our models. Binary classification typically adopts the accuracy measurement, defined as:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{2.5.1}$$

where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives respectively. For multi-class classification, where each input sample is classified as one of many classes, the F_1-score is typically used. It is defined as:

$$\text{F}_1\text{-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{2.5.2}$$

where Precision and Recall are defined as:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \end{aligned} \tag{2.5.3}$$

For machine translation, it is typical to use the BLEU (bilingual evaluation understudy) score to evaluate the quality of the translated text [55]. It is designed to measure the correspondence between the output of a machine learning model and the golden translation. BLEU was one of the first metrics to obtain a high correlation with human judgments of quality,

DOI:10.6814/NCCU202001426

and remains one of the most popular automatic as well as cost-effective metrics. Evaluation of the score involves calculating the similarity between pieces of the translation individually and the reference text. Then, the overall score is the mean over the entire corpus. It should be noted that grammaticality, intelligibility, semantic quality cannot be evaluated in this manner. The value of BLEU score lies between 0 and 1, where 1 indicates that the automatic translation is identical to a segment in the golden translations.

For automatic summarization, we adopt the commonly used "Recall-Oriented Understudy for Gisting Evaluation" (ROUGE) scores [44]. The ROUGE method aims at calculating the ratio of the unit overlap between the generated results and the golden summary. The unit used here can be N-gram or character sequences. Specifically, we use three ROUGE calculation methods: ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (Longest Common Subsequence) scores. To improve legibility, we will abbreviate them as R-1, R-2 and R-L henceforth. Intuitively, R-1 can be thought of as to represent the amount of information of automatic summaries, whereas R-2 is to evaluate the overall fluency of said summaries. Finally, R-L can be regarded as the coverage rate of the summary over the original article.

DOI:10.6814/NCCU202001426

Chapter 3

Methods

This chapter provides descriptions of various neural architectures and how to adapt them for the downstream tasks, which include sequence classification, part-of-speech tagging, named entity recognition, sentiment analysis, entailment, translation, and summarization.

3.1 Neural Networks

Artificial neural networks (ANN) have become a prominent tool for natural language processing in recent years. As a result, a wide variety of network structures are used as the basis of models for the experiments in this work. Before going into details of each task, the model architectures are briefly described in the following sections.

3.1.1 Recurrent Neural Networks

In essence, natural language inputs consist of a sequence of words or sub-word tokens, in which the order cannot be freely alternated. Therefore, RNNs emerge as an intuitive choice.

We propose an approach for identifying protein-protein interaction (PPI) in biomedical

DOI:10.6814/NCCU202001426

literature using RNN with LSTM cells. We employ a straightforward extension named Bidirectional RNN, which encodes sequential information in both directions (forward and backward) and concatenate the final outputs. In this way, the output of one time step will contain information from its left and right neighbors.

For classification tasks including sentiment analysis and entailment detection, we use a Bidirectional LSTM [31] with an attention [4] layer as the sentence encoder, and a fully connected layer for the classification task. Similarly, for tasks such as POS and NER where the label of one character can be determined by its context, bidirectional learning can be beneficial.

For machine translation, we employ a common seq2seq model [65], in which both the encoder and decoder are a 2-layer stacked Bi-LSTM with 512 hidden units.

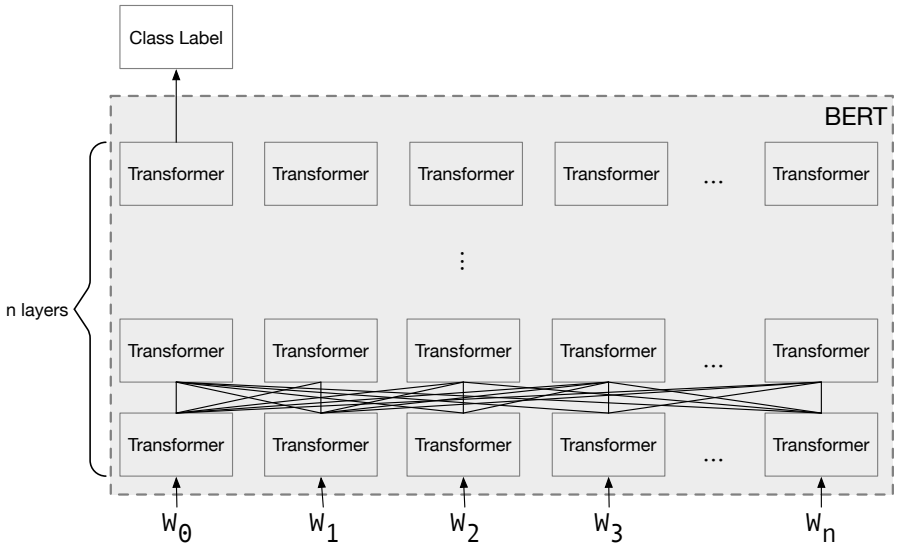
For abstractive summarization, we use a layer of LSTM network with attention mechanism, and compare the difference between uni-directional and bi-directional networks, as well as the impact of the LSTM cell dimension, word vector dimension and other parameters.

3.1.2 Self-Attentive Models

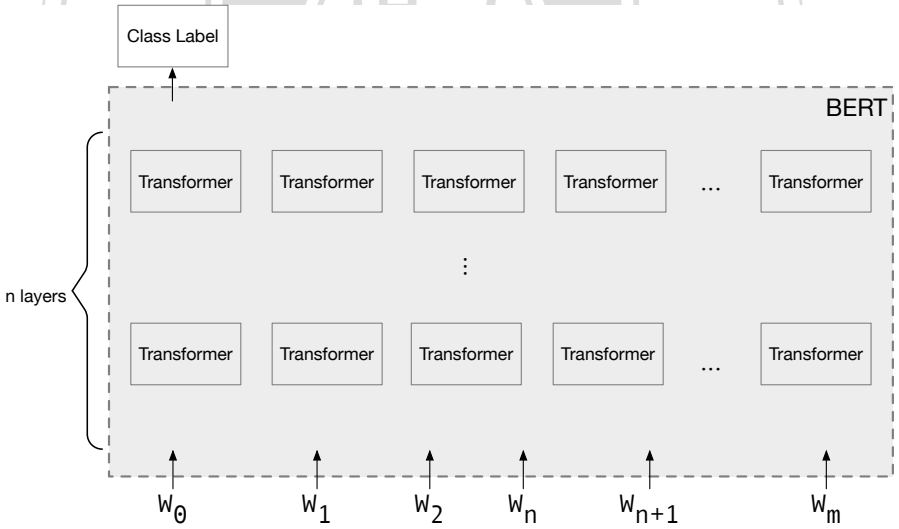
Self-attentive models including Transformer [70] and "Bidirectional Encoder Representations from Transformers," shortened as *BERT* [18], rely on the attention mechanism [46] to learn a context-dependent representation, or encoding. As such, self-attention has been successfully applied in several tasks. Similar to bidirectional LSTM, this type of encoder takes $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_n$ as the input, and produces context-aware word representations $\mathbf{r}_i$ of all positions $0 \leqslant i \leqslant n$. We employ a stack of N identical self-attention layers, each having independent parameters.

The classification problems adopt the BERT model with an identical setup to the original paper, in which BERT is used as an encoder that represents a sentence as a vector. This vector is then used by a fully connected neural network for classification. Note that models are tuned

DOI:10.6814/NCCU202001426



(a) Single sentence classification



(b) Sentence pair classification

Figure 3.1: Classification of sentence and sentence pair using BERT

DOI:10.6814/NCCU202001426

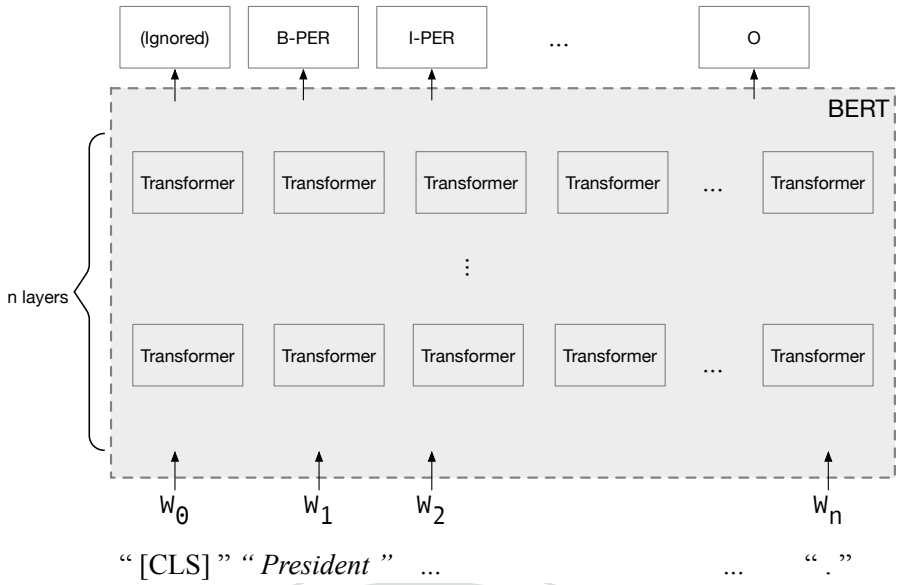


Figure 3.2: Named entity recognition using BERT

separately for each task. Figure 3.1 denotes how to model BERT model for classification tasks. Figure 3.2 denotes how to perform sequence labeling, such as NER, using BERT. Figure 3.3 illustrates the approach for building a question answering system with BERT.

In addition, we tried to determine the effect of pre-training by testing a compact version of BERT, named $\text{BERT}_{\text{NOPT}}$. It comprises of three self-attention layers instead of 12.

To the best of our knowledge, machine translation models do not typically employ BERT. Therefore, for our MT experiments, a Transformer encoder-decoder model is utilized.

3.2 Adversarial Attack Methods

In order to test the robustness of various neural models, we include five methods for generating adversarial examples (attacks). These methods have a common goal, which is to find and replace only one of the elements in an input sequence such that the prediction of the model is incorrect. They are introduced in this section.

The first method is based on random word replacement, which serves as the baseline. The

DOI:10.6814/NCCU202001426

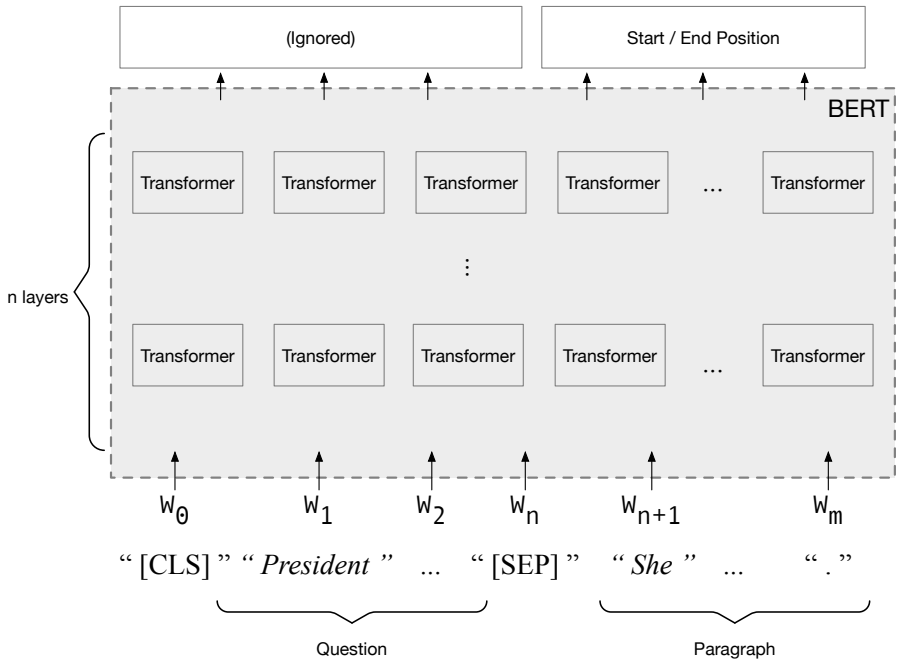


Figure 3.3: Question answering using BERT

second (list-based) and third (greedy) methods are adapted from previous work [3, 14]. The fourth (constrained greedy) and fifth (attention-based) are proposed by the current work.

3.2.1 Random Attack

This basic attack method selects a word in the original sequence and replaces it with another one in the vocabulary, both randomly. In order to fairly estimate the effect of randomness, this attack is repeated for 10^5 trials and averaged to obtain the overall score. It is denoted as **RANDOM**.

3.2.2 List-based Attack

The second method is recently proposed by Alzantot et al. [3], denoted as **LIST**. LIST employs a list of semantically similar words (*i.e.*, *synonyms*), and manages to replace a word in the input sentence with another from the list to construct adversarial examples. In other words,

DOI:10.6814/NCCU202001426

Table 3.1: Illustrative examples of semantically similar words.

Word	Similar Words
abandon	forgo, renounce, relinquish, forego, forswear, forsake, abdicate, waive, abandons, abandoning, renounces
abate	lessen, downsize, reduce, shortening, mitigate, mitigating, reducing, mitigation, curtail, lighten, alleviate, minimize, shorten
	. . .
zucchini	spinach, broccoli, eggplant, celery, leeks, onion, artichokes, cauliflower, tomatoes, chard, eggplants, sauteed, tomato, artichoke, courgettes, radishes, shallots, okra, arugula, beets

the list is used to replace a word with one of its synonyms; this process is repeated for every word in the input sentence until the target model makes an incorrect prediction. That is, for every sentence, we start by replacing the first word with its synonyms, each forming a new adversarial example. If none of these successfully misleads the model, we move to the next word (and the first word remains unchanged), and repeat this process until either the attack succeeds or all words have been tried.

A list of semantically similar words can be found in Table 3.1. We can see that this list enables us to find very closely related words (synonyms) to perform attacks.

3.2.3 Greedy Select & Greedy Replace

The third method (denoted as GS-GR) greedily searches for the weak spot of the input sentence [77] by replacing each word, one at a time, with a "padding" (a zero-valued vector) and examining the changes of output probability. After determining the weak spot, GS-GR then replaces that word with a randomly selected word in the vocabulary to form an attack. This process is repeated until the attack succeeds or all words in the vocabulary are exhausted.

DOI:10.6814/NCCU202001426

3.2.4 Greedy Select with Embedding Constraint

Although the GS-GR method potentially achieves a high success rate, the adversarial examples formed by GS-GR are usually unnatural; sometimes GS-GR completely changes the semantics of the original sentence by replacing the most important word with its antonym, for example: changing "this is a **good** restaurant" into "this is a **bad** restaurant." This cannot be treated as a successful attack, since humans will notice the change and agree with the model's output. This is because GS-GR only considers the classification loss when finding the replacement word, and largely ignore the actual semantics of the input sentence.

To resolve this issue, we propose to add a constraint on sentence-level (not word-level) embedding: the attack must find a word with the minimum L1 distance between two embeddings (from the sentences before and after the word change) as the replacement.

This distance constraint requires a replacement word not to alter the sentence-level semantics too much. This method is denoted as GS-EC. In the experimental results, we show that the GS-EC method achieves a similar success rate as GS-GR in misleading the model, while being able to generate more natural and semantically-consistent adversarial sentences.

3.2.5 Attention-based Select

We conjecture that self-attentive models rely heavily on attention scores, and changing the word with the highest or lowest attention score could substantially undermine the model's prediction. Therefore, this attack method exploits and also investigates the attention scores as a potential source of vulnerability. This method first obtains the attention scores and then identifies a target word that has the highest or lowest score. Target word is then replaced by a random word in the vocabulary, and this process is repeated until the model is misled by the generated

DOI:10.6814/NCCU202001426

adversarial example. These methods are denoted as **AS_{MIN}-GR** that replaces the word with the lowest score, and **AS_{MAX}-GR** with the highest score.

Furthermore, the constraint on the embedding distance can also be imposed here for finding semantically similar adversarial examples; these methods are referred as **AS_{MIN}-EC** and **AS_{MAX}-EC**, respectively. As a pilot study, we examine the attention scores on the first and last layers of the BERT model for understanding the model's behavior under attacks.

DOI:10.6814/NCCU202001426

Chapter 4

Experiments

4.1 Text Sequence Classification in Biomedical Literature

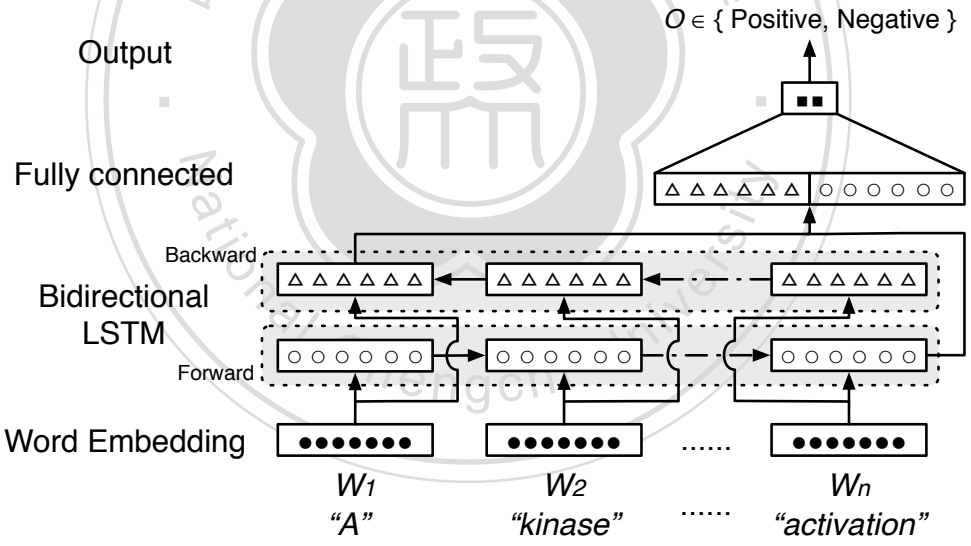


Figure 4.1: Recurrent neural network-based PPI classification model.

This experiment concerns with evaluating an approach to the problem of classifying the textual description of protein-protein interaction (PPI) in biomedical literature. It is one of the essential parts of this field, especially because it can serve as the basis of building a knowledge base and/or ontology for the entities such as molecules and cells within the sentence [52]. The

DOI:10.6814/NCCU202001426

Table 4.1: Descriptive Statistics of AIMed and BioInfer, the two largest PPI corpora.

Corpus	Number of Sentences	Number of Positive/Negative Protein Pairs
AIMed	1,955	1,000 / 4,834
BioInfer	1,100	2,534 / 7,132

rapid growth of the amount of research papers in the world strengthens the need for this task, and newer methods are much in demand.

Here we based on RNNs to capture the long-term relationships among words in order to identify PPIs. The proposed model is evaluated on two largest PPI corpora, *i.e.*, AIMed [6] and BioInfer [58] using cross-validation (CV) and cross-corpus (CC) settings. Figure 4.1 illustrates the structure of this neural network. The descriptive statistics of the datasets used in this experiment is listed in Table 4.1

We adopt 10-fold cross-validation (CV) and cross-corpus (CC) testing schemes for evaluation. The evaluation metrics are the precision, recall, and F1-score for both schemes.

Compared methods include the shortest dependency path-directed constituent parse tree (SDP-CPT) method [59], in which the tree representation generated from a syntactic parser is refined by using the shortest dependency path between two entity mentions derived from a dependency parser; A knowledge-based approach PIPE [10] that extracts linguistic interaction patterns and learned by a convolution tree kernel; A composite kernel approach (CK) [51] which combines several different layers of information from a sentence with its syntactic structure by using several parsers; and a graph kernel method (GK) [2] that integrates parse structure sub-graph and a linear order sub-graph. We further compare with recent NN-based approaches: sdpCNN [34] which combines CNN with shortest dependency path features; McDepCNN [57] that uses positional embeddings along with word embeddings as the input, and a tree kernel using

DOI:10.6814/NCCU202001426

various word embeddings labeled as TK+WE [42]. We also evaluate the effect of pre-training of word embeddings by comparing randomly initialized and pre-trained embeddings, labeled as $\text{LSTM}_{\text{rand}}$ and LSTM_{pre}, respectively.

4.1.1 Experimental Setup

To ensure the generalization of the learned model, the protein names recognized in the text are replaced with "PROTEIN1", "PROTEIN2", or "PROTEIN", where "PROTEIN1" and "PROTEIN2" are the pair of interest, and other non-participating proteins are marked as "PROTEIN". An example is given as follows. The sentence "Thymocyte activation induces the association of phosphatidylinositol 3-kinase and pp120 with CD5" contains three proteins, namely, "phosphatidylinositol 3-kinase", "pp120", and "CD5". In the three possible pairs of proteins, two of them are in interaction relations. Therefore, there are three variants of this sentence with proteins being replaced by the special labels in the data, and two of them are marked as "positive" while the other one as "negative". During testing, all the variants will be scored. The maximum sentence length is set to 100, where longer sentences are truncated and shorter sentences padded with zeros. We use 200-dimension embeddings and 400-dimension LSTM cells. The dropout rate is set to 0.5. The `RMSProp` optimizer [68] with default learning rate settings are applied. We implement the model using `keras` with `tensorflow` [1] backend [1]. With a batch size of 16, training one epoch on one Titan X GPU takes approximately one minute. The throughput of the inference stage is around 130 KiB of text per second.

[1] Code can be downloaded from https://github.com/ylhsieh/ppi_lstm_rnn_keras.

DOI:10.6814/NCCU202001426

Table 4.2: Ten-fold cross-validation of the performance of various PPI classification models on corpora AIMed and BioInfer. Metrics are precision, recall, and F-score (in %) and standard deviation in parentheses. The bold numbers highlight the best performance of a column.

Method	AIMed			BioInfer		
	Precision	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
GK	52.9	61.8	56.4	56.7	67.2	61.3
SDP-CPT	59.1	57.6	58.1	-	-	62.4
CK	55.0	68.8	60.8	65.7	71.1	68.1
PIPE	57.2	64.5	60.6	68.6	70.3	69.4
McDepCNN	67.3	60.1	63.5	62.7	68.2	65.3
sdpCNN	64.8	67.8	66.0	73.4	77.0	75.2
TK+WE	-	-	69.7	-	-	74.0
$LSTM_{rand}$	70.1 (6.5)	70.4 (6.4)	70.1 (5.5)	83.6 (2.4)	83.3 (2.7)	83.4 (2.3)
$LSTM_{pre}$	**78.8** (5.6)	**75.2** (5.0)	**76.9** (4.9)	**87.0** (2.3)	**87.4** (2.3)	**87.2** (1.9)

4.1.2 Results

Ten-fold cross-validation results on AIMed and BioInfer are listed in Table 4.2. Kernel-based methods can achieve decent F-scores of 61% and 69%. All NN-based methods outperform kernel-based ones by up to 10% on AIMed and 5% on BioInfer. When using randomly initialized embeddings, RNN exhibits similar performance as other NN models. However, by taking advantage of pre-trained embeddings, RNN further advances F-scores by 7% and 13% on AIMed and BioInfer, respectively. In other words, pre-training contribute to relative improvements of 10% and 18%. These results demonstrate that, even though kernel-based methods all include syntactic or semantic structures and carefully crafted features, neural networks are capable of automatically capturing contextual information that is crucial for identifying PPIs. Moreover, we can see that the standard deviations of the performance by RNN on the larger corpus,

DOI:10.6814/NCCU202001426

i.e., BioInfer, is much lower than that of the smaller corpus (5 vs. 2). Besides, the relative improvement of RNN over compared methods on BioInfer is greater as well (10% and 18%). This suggests that richer context information in a larger corpus may be difficult to be manually modeled via feature engineering or rule creation, but is a well-suited learning source for neural networks.

Table 4.3: Cross-corpus evaluation of the F-score (in %) of various PPI classification models on corpora AIMed and BioInfer. The bold numbers highlight the best performance of a column.

Method	**Train** on AIMed, **Test** on BioInfer	**Train** on BioInfer, **Test** on AIMed
GK	47.1	47.2
CK	53.1	49.6
PIPE	**58.2**	**52.1**
McDepCNN	48.0	49.9
LSTM	*49.3*	*50.7*

Table 4.3 shows the cross-corpus results, in which different methods are trained on one corpus and tested on the other. We observe that, although RNN performs slightly better than McDepCNN, CK and PIPE methods are much more robust when learning on different corpora. We postulate that knowledge may play an important role in this scenario, and effective incorporation of such information into RNN can be a promising direction for future research.

DOI:10.6814/NCCU202001426

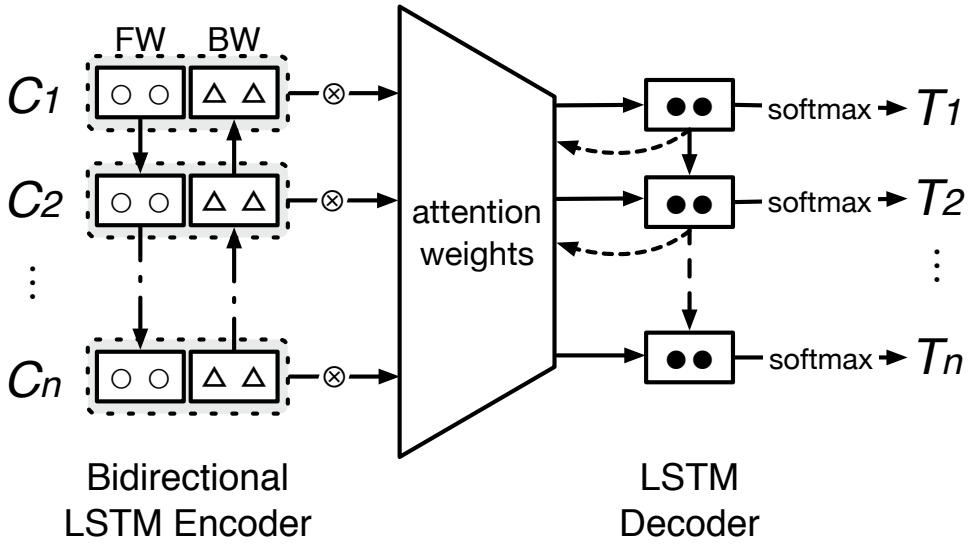


Figure 4.2: Architecture of MONPA: multi-objective named entity & POS annotator.

4.2 Sequence Labeling

This experiment regards using an encoder-decoder [65] structure with the attention mechanism [46] to perform sequence labeling with multi-task objectives. In particular, we conduct Chinese word segmentation, part-of-speech (POS), and named entity (NE) labeling simultaneously. The input is a sequence of Chinese characters that may contain named entities, and the output is a sequence of POS tags and possibly NEs in the form of '`BIES`' tags.

The model that are used in this task mainly consists of: embedding layer, recurrent encoder layers, attention layer, and decoder layers. The embedding layer converts characters into embeddings [50], which are dense, low-dimensional, and real-valued vectors. They capture syntactic and semantic information provided by its neighboring characters. In this work, we utilize pre-trained embeddings using `word2vec` and over 1 million online news articles. The recurrent encoder layers use LSTM cells, which have been shown to capture long-term dependencies in the input sequence.

We employ a straightforward extension named Bidirectional RNN [26], which encodes sequential information in both directions (forward and backward) and concatenate the final

DOI:10.6814/NCCU202001426

outputs. In this way, the output of one time step will contain information from its left and right neighbors. For tasks such as POS and NER where the label of one character can be determined by its context, bidirectional learning can be beneficial. The attention layer is proposed by Luong et al. [46] in an attempt to tackle the problem of finding corresponding words in the source and target languages when conducting machine translation. Finally, the recurrent decoder layers take the sequence of output from the attention layer and project them onto a V-dimensional vector where V equals the number of possible POS and NE tags. The overview of the complete model is shown in Figure 4.2. The loss of the model is defined as the averaged cross-entropy between an output sequence and true label sequence.

Test corpora from five previous SIGHAN shared tasks, which have been widely adopted for Traditional Chinese word segmentation and NER, were used to evaluate the proposed system. Besides the participating systems in the above shared tasks, we also compare with existing word segmentation toolkits Jieba and CKIP [32]. The word segmentation datasets were taken from SIGHAN shared tasks of years 2003–2008, and NER dataset is from 2006. We follow the standard train/test split of the provided data, where 10,000 sentences of the training set are used as the validation set. Details of the word segmentation and NER datasets are shown in Table 4.4 and 4.5, respectively. Three metrics are used for evaluation: precision, recall, and F_1-score. For word segmentation, a token is considered to be correct if both the left and right boundaries match those of a word in the gold standard. For the NER task, both the boundaries and the NE type must be correctly identified.

4.2.1 Experimental Setup

In order to obtain multi-objective labels of the training data, we first merge datasets from the 2006 SIGHAN word segmentation and NER shared tasks. Since rich context information is

DOI:10.6814/NCCU202001426

Table 4.4: Statistics of the number of words in two word segmentation datasets.

Year	AS		CityU	
	Train	**Test**	**Train**	**Test**
2003	5.8M	12K	240K	35K
2005	5.45M	122K	1.46M	41K
2006	5.5M	91K	1.6M	220K
2008	1.5M	91K	-	-

Table 4.5: Statistics of the number of words in the 2006 NER dataset.

#Train/Test Words		
Person	**Location**	**Organization**
36K / 8K	48K / 7K	28K / 4K

able to benefit deep learning-based approach, we augment the training set by collecting online news articles[2]. There are three steps for annotating the newly-created dataset. We first collect a list of NEs from Wikipedia and use it to search for NEs in the corpus, where longer NEs have higher priorities. Then, an NER tool [75] is utilized to label NEs. Finally, CKIP is utilized to segment and label the remaining words with POS tags. Three variants of the proposed model are tested, labeled as RNN_{CU06}, RNN_{YA}, and $\text{RNN}_{\text{CU06+YA}}$. RNN_{CU06} is trained using only word segmentation and NER datasets from the 2006 City University (CU) corpus; RNN_{YA} is trained using only online news corpus, and $\text{RNN}_{\text{CU06+YA}}$ is trained on a combination of the above corpora.

We implemented the RNN model using `pytorch`[3]. The maximum sentence length is set to 80, where longer sentences were truncated and shorter sentences were padded with zeros. The forward and backward RNN each has a dimension of 300, identical to that of word embeddings.

[2]News articles are collected from the Yahoo News website and contains about 3M words.
[3]https://github.com/pytorch/pytorch

DOI:10.6814/NCCU202001426

There are three layers for both encoder and decoder. Dropout layers exist between each of the recurrent layers. The training lasts for at most 100 epochs or when the accuracy of the validation set starts to drop.

4.2.2 Results and Discussion

Note that since we combined external resources, performances of the compared methods are from the open track of the shared tasks. Table 4.6a lists the results of the RNN-based models and top-performing systems for the word segmentation subtask on the Academia Sinica (AS) dataset. First of all, RNNs exhibit consistent capabilities in handling data from different years and is comparable to the best systems in the competition. In addition, it is not surprising that the RNN_{YA} model perform better than RNN_{CU}. Nevertheless, our method can be further improved by integrating the CU06 corpus, demonstrated by the results from the $\text{RNN}_{\text{CU06+YA}}$ model. This indicates that RNN can easily adapt to different domains with data augmentation, which is an outstanding feature of end-to-end models. As for the CU dataset listed in Table 4.6b, all of the RNN models show considerable decrease in F-score. We postulate that it may be due to the training data, which is processed using an external tool focused on texts from a different linguistic context. It is also reported by [75] that segmentation criteria in AS and CU datasets are not very consistent. However, by fusing two corpora, the $\text{RNN}_{\text{CU06+YA}}$ can even surpass the performances of CKIP. Finally, comparison with Jieba validates that the RNN model can serve as a very effective toolkit for NLP researchers as well as the general public.

Table 4.7 lists the performances of proposed models and the only system that participated in the open track of the 2006 SIGHAN NER shared task. We can see that RNN_{CU06} outperforms the model from Yu et al. [78], confirming RNN's capability on jointly learning to segment and recognize NEs. Interestingly, RNN_{YA} obtains a much lower F-score for all NE types. And

DOI:10.6814/NCCU202001426

Table 4.6: Word segmentation performance (% F-score) of various systems on different years of SIGHAN shared tasks, split into the Academia Sinica (AS) and City University (CU) datasets. The best performance in a column is marked bold.

(a) AS dataset, open track

System	F-score			
	2003	2005	2006	2008
Gao et al. [22]	95.8			
Yang et al. [76]	90.4			
Low et al. [45]		**95.6**		
Chen et al. [11]		94.8		
Zhao et al. [81]			**95.9**	
Jacobs and Wong [36]			95.7	
Wang et al. [71]			95.3	
Chan and Chong [9]				**95.6**
Mao et al. [49]				93.6
Jieba	83.0	80.9	81.3	81.8
CKIP	**96.6**	94.2	94.6	94.9
RNN_{CU06}	88.4	86.8	87.1	87.4
RNN_{YA}	94.4	92.8	93.0	93.3
$RNN_{CU06+YA}$	*94.6*	*93.2*	*93.6*	*93.8*

(b) CU dataset, open track

System	F-score		
	2003	2005	2006
Ma and Chen [47]	**95.6**		
Gao et al. [22]	95.4		
Peng et al. [56]	94.6		
Yang et al. [76]	87.9		
Low et al. [45]		**96.2**	
Chen et al. [11]		94.5	
Zhao et al. [81]			**97.7**
Wang et al. [71]			97.7
Jacobs and Wong [36]			97.4
Jieba	80.3	81.2	82.4
CKIP	89.7	89.0	89.8
RNN_{CU06}	87.6	85.8	87.8
RNN_{YA}	88.0	87.2	88.5
$RNN_{CU06+YA}$	*91.5*	*90.1*	*91.7*

$RNN_{CU06+YA}$ can only obtain a slightly better F-score for person recognition but not the overall performance of RNN_{CU06}, even with the combined corpus. We believe that boundary mismatch may be a major contributing factor here. We also observe that there are a large number of one-character NEs such as abbreviated country names, which can not be easily identified using solely character features.

DOI:10.6814/NCCU202001426

Table 4.7: NER performance (% F-score) of different systems on the 2006 SIGHAN NER shared task (open track). The best performance in a column is marked bold.

System	F-score			
	PER	LOC	ORG	Overall
Yu et al. [78]	80.98	86.04	68.01	80.51
RNN_{CU06}	81.13	**86.92**	**68.77**	**80.68**
RNN_{YA}	70.54	67.80	31.35	52.62
$RNN_{CU06+YA}$	**83.01**	82.46	54.57	75.28

4.3 Sentiment Analysis

Sentiment analysis attempts to detect the information regarding the attitude or view of a piece of text, through the implementation of some computational methods. For example, one may want to know the emotion that is expressed in a short online product review. Unlike using statistics of the surface words to successfully determine the topic or category of a news article, sentiment analysis may be faced with more challenges. As we can imagine, a positive review may contain positive keywords such as "best" or "awesome," but a negative one can have absolutely no such clue.

In this experiment, we evaluate the robustness of LSTM, BERT, and $BERT_{NOPT}$ models on sentiment analysis using the Yelp dataset [80]. This dataset includes predefined training and test sets. Models under attack have accuracies of 93.7%, 87.3% and 90.7% for fine-tuned BERT model, $BERT_{NOPT}$ and LSTM, respectively, on the test set.

For attention-based attacks (*i.e.*, AS_{MIN}-GR, AS_{MAX}-GR, AS_{MIN}-EC, and AS_{MAX}-EC), the average of the first (*i.e.*, the one that is closest to the model input) attention layer from all 12 heads in BERT and $BERT_{NOPT}$ are used for our attacks. It is worth mentioning that, as a pilot study, we tested using the last layer during AS_{MAX}-EC attack. However, experimental results

DOI:10.6814/NCCU202001426

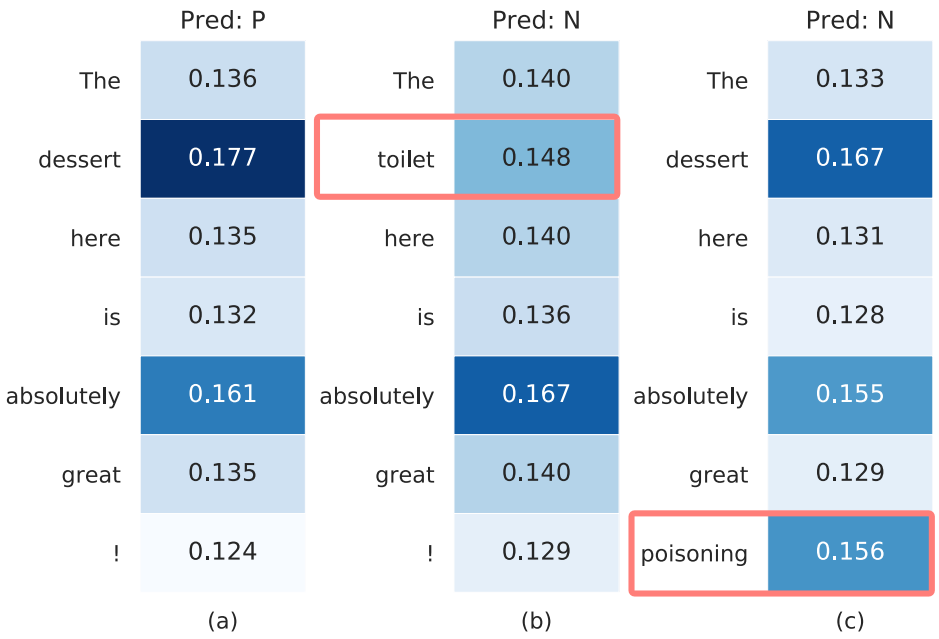


Figure 4.3: Comparison of the distribution of attention scores in a model when the input is (a) the original input, (b) AS_{MIN}-EC, and (c) AS_{MAX}-EC attacks. The word that is selected by the attack is indicated by red boxes. Note how the selection of the target word is based on the lowest or highest attention score, as defined by AS_{MIN}-EC, and AS_{MAX}-EC. Both attacks successfully changed the prediction of the model from positive to negative.

exhibited a $< 10\%$ success rate. Therefore, only the results from using the first attention layer are included.

4.3.1 Results

To illustrate how adversarial attacks work, Fig. 4.3 shows the results from AS_{MAX}-EC and AS_{MIN}-EC methods that select a word to change based on the attention scores of the original sentence.

A comprehensive quantitative comparison can be found in Table 4.8, from which we make the following observations.

1. Greedy-based attacks consistently achieve higher successful rate than other attacks. The proposed GS-EC method can achieve almost identical success rates with GS-GR while

DOI:10.6814/NCCU202001426

Attack Method	Model		
	LSTM	**BERT**	**$\text{BERT}_{\text{NOPT}}$**
RANDOM	1.1%	0.8%	1%
LIST	27%	6%	15%
$\text{AS}_{\texttt{MIN}}$-GR	16%	11%	32%
$\text{AS}_{\texttt{MAX}}$-GR	62%	17%	35%
$\text{AS}_{\texttt{MIN}}$-EC	16%	10%	32%
$\text{AS}_{\texttt{MAX}}$-EC	62%	17%	35%
Best attention attack(A_*)	62%	17%	35%
GS-GR	**79%**	**52%**	**53%**
GS-EC	*78%*	*50%*	**53%**

Table 4.8: Effectiveness (% success) of different attacks on sentiment analysis models. The highest attack rate in a column is marked bold.

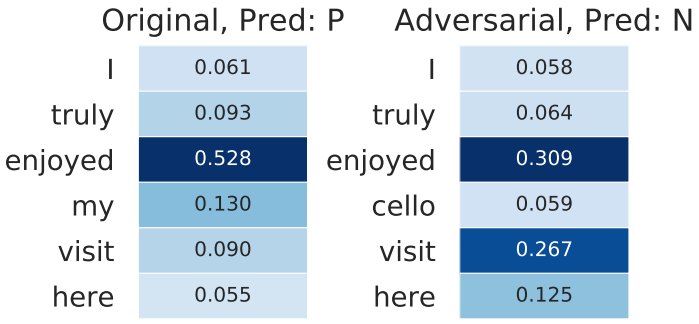
restricting the search space based on the embedding distances. We will further show that GS-EC leads to higher quality adversarial examples in Section 4.3.2.

2. We found that using attention, especially **$\text{AS}_{\texttt{MAX}}$-GR** and **$\text{AS}_{\texttt{MAX}}$-EC** methods, can easily break the LSTM model. However, the same vulnerability does not exist in BERT or $\text{BERT}_{\text{NOPT}}$ models. Since different types of attention-based attacks are suitable for different models, we summarize the best attention-based attack performance as A_* in the table, which takes the maximum over four different types of attention-based attacks.

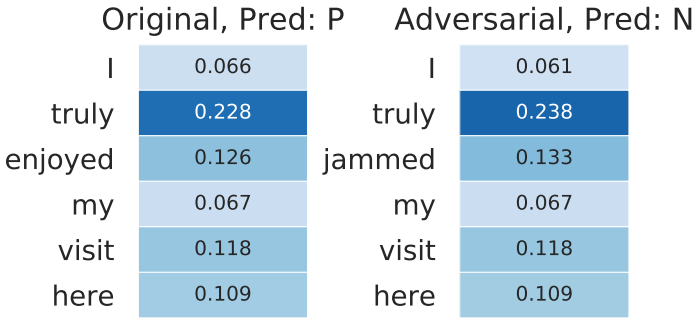
3. Self-attentive models (BERT and $\text{BERT}_{\text{NOPT}}$) consistently lead to lower attack successful rates compared with the LSTM model, under RANDOM, LIST, attention-based attacks and greedy-based attacks.

We demonstrate the robustness of BERT model under GS-EC attack in Fig 4.4. By

DOI:10.6814/NCCU202001426



(a) LSTM



(b) BERT

Figure 4.4: Shift of attention scores under GS-EC attack on (a) LSTM and (b) BERT models.

observing the shift in attention when a model is under attack, we can infer the strength of different models. It is shown that, although the GS-EC attack successfully changes the sentiment prediction from positive to negative for both models, the distribution of attention scores of the models are different. In particular, scores remain stable for BERT model, whereas the LSTM suffers from a large shift in attention distribution.

4.3.2 Quality of Adversarial Examples

We conduct experiments to assess the naturalness of adversarial examples. First, Table 4.9 compares the quality of the results generated by GS-GR and GS-EC attacks on a BERT model. Here we see that constraints imposed by GS-EC make it superior than GS-GR in terms of retrieving words that are coherent with the context.

DOI:10.6814/NCCU202001426

Method	Sentence
GS-GR	Pizzeria Bianco was a such **never** a nice treat that was [...]
GS-EC	Pizzeria Bianco was a such **ostensibly** a nice treat that was [...]
GS-GR	The desserts here are absolutely great **0** ! [...]
GS-EC	The desserts here are absolutely great **soluble** ! [...]

Table 4.9: Example of adversarial attacks on BERT sentiment analysis models, as generated by GS-GR and GS-EC approaches. These attacks can change the output of the model such that the opposite sentiment is predicted. Notably, attacks by GS-EC utilize words that are locally coherent and fluent, possibly due to the constraint on embedding similarity. On the other hand, GS-GR attacks are more incoherent.

Furthermore, we organize a large-scale human evaluation on Amazon Mechanical Turk regarding the qualities of adversarial examples generated by different methods. Each sample is voted by 3 turkers. Recall that, we previously defined "Readability" and "Human accuracy." The former is regarded as the relative naturalness of the adversarial examples (normalized to the maximum between the two), and the latter is the percentage of human responses that matches the true label.

Table 4.10 is a comparison of LSTM and BERT models using the GS-EC attack. It shows that the distance in embeddings space of BERT can better reflect semantic similarity and contribute to more natural adversarial examples. And, in Table 4.11, we compare using GS-GR and GS-EC method on BERT model. Again, we see that the GS-EC method, which restricts the distance between sentence embeddings of original and adversarial inputs, can produce superior adversarial examples.

DOI:10.6814/NCCU202001426

Model	Readability	Human Accuracy
LSTM	0.6	52.1%
BERT	**1.0**	**68.8%**

Table 4.10: Human evaluations of the quality of attacks on LSTM and BERT models using GS-EC attack.

Method	Readability	Human Accuracy
GS-GR	0.55	64.6%
GS-EC	**1.0**	**68.8%**

Table 4.11: Human evaluations of GS-GR and GS-EC attacks on BERT model for sentiment analysis.

4.4 Textual Entailment

We conduct evaluations on MultiNLI [74] dataset for textual entailment with approaches similar to the ones in the last section. MultiNLI is one of the many datasets that see major improvements by BERT. The BERT model is trained to achieve 83.5% accuracy and LSTM 76%. $\text{BERT}_{\text{NOPT}}$ is excluded from this experiment since it cannot reach a satisfactory accuracy.

4.4.1 Results

Results from entailment models fall into the same pattern as those from sentiment analysis, which is listed in Table 4.12.

Our findings are summarized as follows:

1. The entailment task is more difficult than single-sentence classification, as evidenced by the higher success rates of attacks among all models and attacks.

2. The greedy-based attacks consistently achieve higher success rates.

DOI:10.6814/NCCU202001426

	Model	
Attack Method	**LSTM**	**BERT**
RANDOM	17.8%	9.2%
LIST	63%	56%
AS_{MIN}-GR	57%	53%
AS_{MAX}-GR	78%	54%
AS_{MIN}-EC	55%	52%
AS_{MAX}-EC	78%	51%
Best attention attack(A_*)	78%	54%
GS-GR	**95%**	**75%**
GS-EC	**95%**	**75%**

Table 4.12: Rate of success (%) of different attacks on LSTM and BERT for the MultiNLI models (dev set). The highest attack rate in a column is marked bold.

3. **AS_{MAX}-GR** and **AS_{MAX}-EC** methods continue to be superior than **AS_{MIN}-GR** and **AS_{MIN}-EC**, although the difference here is not as drastic as in the previous experiment.

4. BERT model remains more robust compared with LSTM.

4.4.2 Quality of Adversarial Examples

Samples illustrated in Table 4.13 show that the GS-EC method can find more coherent words for the attack, as opposed to GS-GR.

For instance, changing the word “great” to “vast” can cause the model to misjudge the entailment relation in the second example. Unfortunately due to budget constraints, we did not conduct large scale human experiments on this dataset.

DOI:10.6814/NCCU202001426

Label	Sentence 1	Sentence 2
Contradiction →Neutral	No, I don't know.	(Original) Yes , I *know*.
		(GS-GR) Yes, I **0**.
		(GS-EC) Yes, I **renovated**.
Neutral→ Contradiction	That's it. The girl looked at him, then passed her hand across her forehead.	(Original) The girl looked at him with *great* interest.
		(GS-GR) The girl looked at him with **!** interest.
		(GS-EC) The girl looked at him with **vast** interest.
Entailment →Neutral	(Original) Workers are also represented in civil rights and *retaliation* claims.	Some workers are represented in civil rights and retaliation claims.
	(GS-GR) Workers are also represented in civil rights and **?** claims.	
	(GS-EC) Workers are also represented in civil rights and **targets** claims.	

Table 4.13: Adversarial examples generated by GS-GR and GS-EC attacks for BERT entailment classifier.

DOI:10.6814/NCCU202001426

Table 4.14: Statistics of the LCSTS corpus.

	Training	**Test**
Number of articles	2,400,591	725
Mean number of chars in an article	103.7	108.1
Mean number of chars in a summary	17.9	18.3

4.5 Abstractive Summarization

Recent work verified the effectiveness of RNNs in abstractive (rewriting) automatic summarization. This section explores the effect of adding attention mechanism into an RNN. The main advantage of the attention mechanism is that it can place higher attention weights on key segments while generating text, thereby composing a more natural summary. In addition, we investigate the difference between uni-directional and bi-directional LSTM neural networks.

The corpus used in this article is the Large-scale Chinese Short Text Summarization dataset (LCSTS) [33]. It is a collection of short text from Sina Weibo, a social media website, posted by a news agency. Each post contains a summary and the body of a brief news story. Descriptive statistics of the LCSTS corpus is listed in Table 4.14.

4.5.1 Experimental Setup

In this experiment, each post in the training data is fed into the model in the form of a sequence of Chinese characters. That is, without word segmentation. The upper limit of the number of characters in the dictionary is the first 4,000 most frequent characters in the corpus, which is consistent with previous studies. We use a layer of LSTM network with attention mechanism, and compare the difference between uni-directional and bi-directional networks, as well as the impact of the LSTM cell dimension, word vector dimension and other parameters.

DOI:10.6814/NCCU202001426

The remaining hyperparameters are set as follows. We select Stochastic Gradient Descent (SGD) as the optimization method, with the learning rate set as 1. The model is trained for at most 20 epochs. We also set a schedule for learning rate decay that will decrease it by 90% for every epoch. The maximum gradient norm is set to be 5. The complete training time of a set of hyperparameter combination is about 48 hours on a single GPU.

4.5.2 Results

The focus of this experiment on generating abstractive summarization is to evaluate the fluency of the summary, which is a very important indicator of readability, so the metrics that we are most interested in is the ROUGE-2 or R-2 scores defined in Section 2.5.

First, we explore the influence of hyperparameters including the dimensions and directionality of the neural network on the quality of the summary. The word vectors are set as 128 or 300 dimensions, and the dimensions of the memory cell in LSTM as 128 or 300. Additionally, unidirectional or bidirectional networks are compared. Note that, due to hardware limitations, we can only select a few combinations of these settings instead of doing an exhaustive search.

The results are shown in Table 4.15. Regardless of whether unidirectional or bidirectional network is used, the increased dimension can bring forth considerable benefits. This indicates that when learning sequential information, the higher-dimension LSTM cell can capture more abundant information. However, we can also find that if the dimension of the LSTM unit is low, increasing the dimension of the word vector will not be beneficial.

In addition, under the same dimension setting, we observe that the effect of using a bidirectional LSTM network is notably better than a unidirectional one. This implies that when generating abstractive summaries, it is necessary to consider a wider context in the original article. In other words, relying on only unilateral information can yield sub-par results. In

DOI:10.6814/NCCU202001426

Table 4.15: Effects of directionality and dimension on ROUGE scores of abstractive summaries using the LCSTS corpus.

Directionality	Dim Word Vec	Dim RNN	R-1	R-2	R-L
Uni	128	128	0.305	0.188	0.280
Uni	128	300	0.328	0.206	0.303
Uni	300	128	0.315	0.193	0.285
Uni	300	300	0.348	0.222	0.320
Bi	128	128	0.324	0.207	0.305
Bi	128	300	0.360	0.235	0.335
Bi	300	128	0.332	0.213	0.311
Bi	300	300	**0.369**	**0.243**	**0.343**

conclusion, the most outstanding automatic summaries can be achieved by using neurons with appropriate dimensions in a bidirectional LSTM network.

We then compare the best performing hyperparameter setting with two methods from previous work. The first is the method proposed by Hu et al. [33], in which a unidirectional RNN without attention mechanism (referred to as HU) is employed. The other one utilized a bidirectional, multilayer RNN with the distraction mechanism [13] (referred to as CHEN). In particular, CHEN compared with the architecture proposed by this research, in addition to the deep neural network and higher dimensions, its distraction mechanism is also more complicated, and the unit used is GRU, which is different from LSTM used in this article.

From the results in Table 4.16, we can first observe that the bidirectional RNN from CHEN's work has greatly improved compared to the baseline method by HU. This again shows that a bidirectional network can effectively learn more contextual information. As for the model proposed in this research, using a simpler attention mechanism than the method by CHEN, combined with a lower-dimensional LSTM, we can exceed the existing state-of-the-art result

DOI:10.6814/NCCU202001426

Table 4.16: Compare ROUGE on the LCSTS corpus using different methods.

Method	Directionality	Dim Word Vec	Dim RNN	R-1	R-2	R-L
Hu et al. [33]	Uni	-	-	0.299	0.174	0.272
Chen et al. [13]	Bi	500	500	0.352	0.226	0.325
Our approach	Bi	300	300	**0.369**	**0.243**	**0.343**

in three different evaluation indicators. Notably, the R-2 score obtains an improvement of more than 2%.

We further postulate that this phenomenon may represent that when the dimension of the neural cells is too high, over-fitting occurs which results in a greater difference between the automatic and golden summary. In addition, it may also be due to the different cell structure (LSTM and GRU) used.

Next, we list some illustrative examples to demonstrate the automatic summaries generated by our model for the readers to make subjective and qualitative analysis. Then, they can verify the correctness and readability of the content.

In order to find exemplar instances, we first sort the generated summaries according to the R-2 score from high to low, and then select those with the highest value (R-2 score higher than 0.8) and the lower (R-2 score equals 0) for this comparison.

Table 4.17 lists some better examples of automatic summaries with R-2 scores higher than 0.8. It can be said that our method can generate legible and fluent summaries that covers important content in the original article. However, parts of the details are sometimes omitted. A case such as "81.4" and "81" in the first example is one of those instances. Another case as in the second example is the inclusion of terms "six" and "jointly", which leads to a slightly lowered R-2 score. Although slight modifications occur, they do not impair our understanding of the essential information in the original text. These instances show that the summarization model

DOI:10.6814/NCCU202001426

proposed in this article can achieve satisfactory results and can indeed create natural abstracts very similar to those written by real human.

Original	正處於風口浪尖的國內奶粉行業出現大交易。蒙牛乳業（02319.HK）以及雅士利（01230.HK）昨日發佈公告稱，蒙牛乳業將斥資 81.4 億港元收購雅士利約 65.4% 股權。業界稱，此舉有助於蒙牛乳業補上奶粉短板，以期重新超越伊利成為行業領頭羊。
Predicted	蒙牛 81 億港元收購雅士利
Answer	蒙牛 81.4 億港元收購雅士利
Original	27 日，六名全國人大代表聯名向全國人大發出建議書，建議取消徵收社會撫養費。建議書認為，徵收社會撫養費，是把「提倡」變成了「強制要求」，侵犯了公民的合法權益。根據不完全統計，全國每年徵收的社會撫養費超過 200 億元。
Predicted	六位 全國人大代表 聯名 建議取消社會撫養費
Answer	全國人大代表建議取消社會撫養費

Table 4.17: Examples of higher-quality abstractive summaries generated by our method. Some differences between the generated and golden summary are marked by underlines.

On the other hand, Table 4.18 lists some examples that are regarded as low quality (R-2 score equals 0). We take a closer look at these cases. First of all, we can see that this model may produce unreasonable abstract sentences (such as the first example in Table 4.18). Although it still seems to be a plausible sentence, the semantics is completely unrecognizable. This kind of error is quite common with current machine-generated language. We speculate that it is due to the fact that the beam search is based on the overall probability, and does not take into account other linguistic characteristics such as grammar or semantics. In the future, we may consider integrating features such as syntactic and semantic intelligibility into model training to avoid such problems. Additionally, we can observe that the lengths of the summary in these examples are too restricted. This is because we normally set it to stop as long as the generated word is “eos”, which indicates “end of sentence,” during the generation. The purpose

DOI:10.6814/NCCU202001426

of this setting is to increase the chance of producing complete sentences, but it may also lead to insufficient length of some sentences. One possible remedy in the future would be to ignore the special symbol and keep the generation as long as possible to achieve better results. Next, we can observe another phenomenon from the second example in Table 4.18. At first glance, we consider the automatic summary to be reasonable, yet the score is low because of advanced writing techniques that reporters and writers usually employ. Such skills including metaphors or idioms transforms the original text into completely different wordings of the summary. The current machine learning models cannot easily capture this unique ability of humans. It also indicates that there is still much room for improvement and development in the field of automatic abstractive summarization.

Original	症狀表現：頭疼胸悶、手心出汗，心緒不寧。常會自言自語：到底選哪款好，怎麼辦，幾款我都好喜歡！小編認為，最重要的第一步是狠狠地深呼吸，除了為使在大戰中頭腦清醒外，還能順便提前「倒吸一口涼氣」，因為您的錢包又要被掏空了。
Predicted	你的錢包好喜歡
Answer	雙十一攻略：當網購狂遇上「選擇困難症」時
Original	李克強此次東歐之行，為中國與中東歐國家傳統友誼的延伸鋪路架橋，為雙方互利共贏的經貿合作穿針引線，為中歐戰略夥伴關係的全面發展添火加柴。經過三年「16+1」機制的運作，當前中國與中東歐的合作成效初顯。
Predicted	李克強與中東歐合作初顯
Answer	地理上的「遠親」心靈上的「近鄰」

Table 4.18: Examples of abstractive summaries with lower ROUGE scores generated by our method.

In the end, we claim that the summary produced by the proposed method has been significantly improved, especially when we look at its coverage and readability. Therefore, we verify that an RNN can successfully identify and produce a satisfactory abstractive summarization.

DOI:10.6814/NCCU202001426

4.6 Machine Translation

We implement LSTM and Transformer machine translation models using OpenNMT [48][4]. Specifically, for the LSTM model, we train it with 453,000 pairs from the Europarl corpus of German-English WMT 15 Task[5], common crawl, and news-commentary. The LSTM model is a two-layer bidirectional LSTM with 512 hidden units together with an attention layer. We use the default hyper-parameters, and reproduce the performance reported by Ha et al. [28]. For the Transformer, we use a public pre-trained model with 6 self-attention layers provided by OpenNMT that reproduces the performance reported by Vaswani et al. [70].

Unlike the classification tasks, in machine translation the attack goal is harder to define. We chose to evaluate the robustness under two types of attacks. In the first type of "targeted keyword attack" discussed in [14], we attempt to generate an adversarial input sequence such that a specific keyword appears in the output sequence within the threshold Δ of number of word changes we allowed. Empirically, we set $\Delta = 3$ in these experiments and adopt the most successful attack, GS-EC, to this case. For the second type of untargeted attack, we consider perturbing the input to degrade the BLUE score of output sequences with respect to the ground-truths. For doing this, we conduct a typo-based attack [5]. Specifically, we randomly select one word in each sentence and change it to a typo predefined in a common typo list. This can be viewed as an extension of LIST attack to the translation task.

4.6.1 Results

For the targeted keyword attack, the success rates on both models are reported in Table 4.19. First, we notice that the success rate of the attacks are below 30%, presumably because

[4] https://github.com/OpenNMT/OpenNMT-py
[5] http://www.statmt.org/wmt15/translation-task.html

DOI:10.6814/NCCU202001426

translation is substantially more complex compared with the aforementioned text classification tasks. Nevertheless, the attacks on the Transformer model is significantly less successful than the LSTM-based one.

For the typo-based attack, the BLUE scores before/ after the attack are reported in Table 4.20. We observe that the Transformer-based model always achieves a higher BLEU score over LSTM-based model, *i.e.*, have a better translation performance whether the sentences contain typos or not. We conclude that Transformer-based model exhibits a greater robustness over LSTM-based model in the case of machine translation. This is consistent with our findings in the previous experiments on sentiment and entailment classification problems.

In addition, we present some successful adversarial examples in Table 4.21, and see that the greedy attack can indeed generate natural examples for both models.

Attack Method	**LSTM**	**Transformer**
GS-EC	27.5%	10.5%

Table 4.19: Success rate of targeted attack on translation models using the GS-EC method.

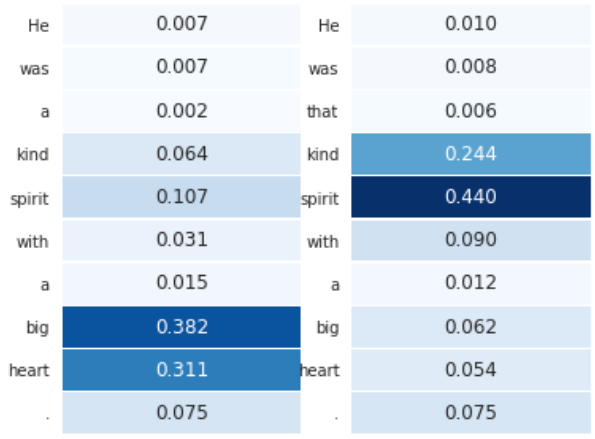
Model	**Original**	**Adversarial**
LSTM	25.10	13.44
Transformer	**34.90**	**26.02**

Table 4.20: Comparison of BLEU scores using typo-based attack on translation models built with LSTM and Transformer models.

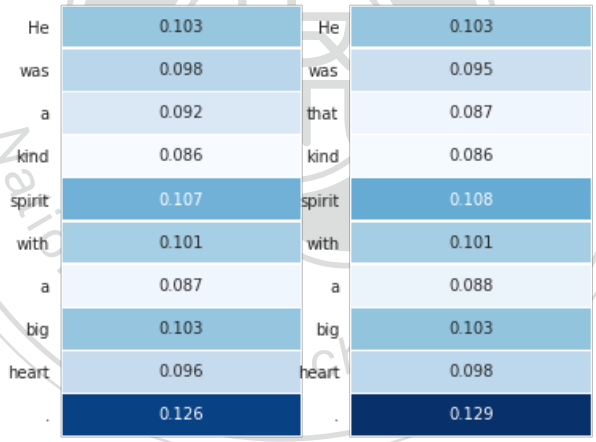
4.7 Summary

We conducted six experiments to evaluate the capabilities of neural networks, including RNNs, BERT, and Transformer. Results show that RNNs can successfully find clues for

DOI:10.6814/NCCU202001426



(a) LSTM



(b) Transformer

Figure 4.5: Heatmap of attention scores in LSTM and Transformer models for machine translation when the input is original and adversarial.

DOI:10.6814/NCCU202001426

LSTM	Org. input	There is a fundamental philosophical reason for the differences between Donald Trump's and Hillary Clinton's [...]
	Adv. input	There is a fundamental philosophical **r** for the differences between Donald Trump's and Hillary Clinton's [...]
	Org. output	Es gibt einen grundlegenden philosophischen Grund für die Unterschiede zwischen Donald Trump und Hillary Clinton s
	Adv. output	Es gibt eine grundlegende philosophischer **Art** , wie Unterschied e zwischen Donald Trump und Hillary Clinton s
Transformer	Org. input	And in this vein , he passed the prize money of 2 5,000 euros on straight away
	Adv. input	And **as** this vein , he passed the prize money of 2 5,000 euros on straight away
	Org. output	Und in diesem Sinne hat er sofort das Preis geld von 2 5.000 Euro über wiesen
	Adv. output	Und als diese **Art** , ging er sofort das Preis geld von 2 5.000 Euro weiter
LSTM	Org. input	There is no clear consensus on where they can seek common ground on Syria
	Adv. input	There is no **"** consensus on where they **she first** common **out** on **year**
	Org. output	Es gibt keine klare Übereinstimmung darüber , wo sie gemeinsame Boden auf Syrien suchen können
	Adv. output	Es gibt keinen " Konsens über die Frage , wo sie sich im Jahr " zunächst auf die selben **Art** befinden
Transformer	Org. input	There is no clear consensus on where they can seek common ground on Syria
	Adv. input	There is no clear consensus on where they can seek common **would it** Syria
	Org. output	Es gibt keinen klaren Konsens darüber , wo sie eine gemeinsame Basis in Syrien suchen können
	Adv. output	Es gibt keinen klaren Konsens darüber , wo sie nach einer gemeinsamen **Art** Syrien suchen können

Table 4.21: Targeted adversarial examples for machine translation models based on LSTM and Transformer (denoted by **TF**) with the target keyword "Art." in the output.

DOI:10.6814/NCCU202001426

identifying PPIs in biomedical literature, the sentiment of reviews, and labeling and segment Chinese words. Using two RNNs as encoder-decoder framework, we can generate natural abstractive summaries of news articles. Transformer-based models can further surpass the performances of RNNs in nearly all of the tasks that we have tested. Moreover, the encoding distance of Transformer-based encoder can be used as the semantic similarity measurement to find similar sentences for the purpose of adversarial attacks.

DOI:10.6814/NCCU202001426

Chapter 5

Discussions

All the above experiments conclude that a self-attentive model exhibits higher robustness compared to a recurrent one. This is somewhat counter-intuitive—at the first glance one may assume that the self-attention layer is not robust since perturbation in one word can affect all the attention scores. We then dive into some theoretical discussions regarding this topic.

5.1 Theoretical Analysis

In this section, we provide some explanation regarding this phenomenon by studying how error propagates through the self-attention architecture. We show that the perturbation of one input embedding can only have sparse effects to the attention scores when the input embedding are scattered enough in the space.

5.1.1 Sensitivity of Self-attention Layer

First, we consider the simple case of one self-attention layer with a single head. Assume a sentence with n input words, and each word is represented by a d-dimensional embedding

DOI:10.6814/NCCU202001426

vectors, denoted by $x_1, \ldots, x_n \in R^d$. And the matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V \in R^{d\times k}$ represent the weights used by query, key, and value transformation calculations. The contribution of each element j to i is then computed by

$$s_{ij} = x_i^T \boldsymbol{W}^Q (\boldsymbol{W}^K)^T x_j, \tag{5.1.1}$$

and then the i-th embedding at the next layer is obtained by

$$z_i = \sum_j \frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}} (\boldsymbol{W}^V x_j), \tag{5.1.2}$$

Sometimes z_i is fed into another linear layer to obtain the embeddings. Now, consider that a small perturbation is added to a particular index $\bar{j}$, so $x_{\bar{j}}$ is changed to $x_{\bar{j}} + \Delta x$ while all the other $\{x_j \mid j \neq \bar{j}\}$ remain unchanged. We study how much will this perturbation affect $\{z_i\}_{i\in[n]}$. For a particular i ($\neq j$), the s_{ij} is only changed by one term since

$$s'_{ij} = \begin{cases} s_{ij} & \text{if } j \neq \bar{j} \\ s_{ij} + x_i^T \boldsymbol{W}^Q (\boldsymbol{W}^K)^T \Delta x & \text{if } j = \bar{j} \end{cases} \tag{5.1.3}$$

where we use s'_{ij} to denote the value after perturbation. Therefore, with the perturbed input, each set of $\{s_{ij}\}_{j=1}^n$ will only have one term that is being changed. Furthermore, the changed term in equation 5.1.3 is the inner product between x_i and a fixed vector $\boldsymbol{W}^Q(\boldsymbol{W}^K)^T\Delta x$; although this could be large for some particular x_i in the similar direction of $\boldsymbol{W}^Q(\boldsymbol{W}^K)^T\Delta x$, if the embeddings $\{x_i\}_{i=1}^n$ are scattered enough over the space, the inner products cannot be large for all $\{x_i\}_{i=1}^n$. Therefore, the change to the next layer embedding will be sparse. For instance, we can prove the sparsity under some distributional assumptions on $\{x_i\}$:

DOI:10.6814/NCCU202001426

Theorem 5.1.1. Assume $\|\Delta x\| \leq \delta$ and $\{x_i\}_{i=1}^n$ are d-dimensional vectors uniformly distributed on the unit sphere, then $E[|s'_{i\bar{j}} - s_{i\bar{j}}|] \leq \frac{C\delta}{\sqrt{d}}$ with $C = \|\boldsymbol{W}^Q\|\|\boldsymbol{W}^K\|$ and $P(|s'_{i\bar{j}} - s_{i\bar{j}}| \geq \epsilon) \leq \frac{C\delta}{\epsilon\sqrt{d}}$.

Proof. The value $E[s'_{i\bar{j}} - s_{i\bar{j}}] = E[x_i^T z]$ where $z = \boldsymbol{W}^Q(\boldsymbol{W}^K)^T\Delta x$ is a fixed vector, and it is easy to derive $\|z\| \leq \|\boldsymbol{W}^Q\|\|\boldsymbol{W}^K\|\delta$. To bound this expectation, we first try to bound $a_1 = E[x_i^T e_1]$ where $e_1 = [1, 0, \ldots, 0]$. Due to the rotation invariance we can get $a_1 = \cdots = a_d$ and $\sum_i a_i^2 = 1$, so $|a_1| = \frac{1}{\sqrt{d}}$. This implies $E[x_i^T z] \leq \frac{C\delta}{\sqrt{d}}$. Using Markov inequality we can then get the probability results. □

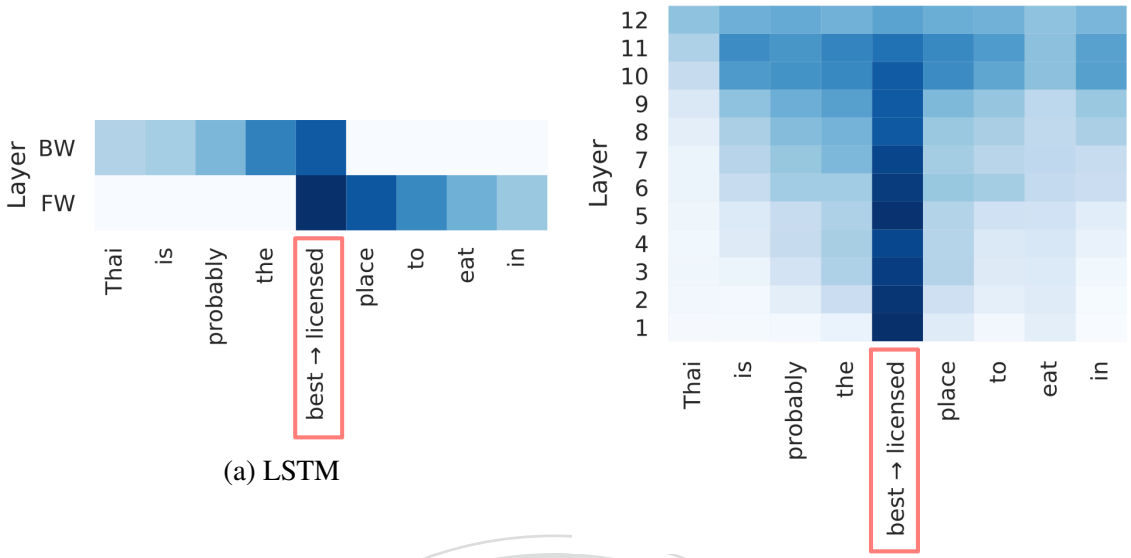
Therefore, as the norm of $\boldsymbol{W}^Q, \boldsymbol{W}^K$ are not too large (usually regularized by L2 during training) and the dimension d is large enough, there will be a significant amount of i such that s_{ij} is perturbed negligibly.

In contrast, embeddings from RNN-based models are relatively more sensitive to perturbation of one word, as shown below. Similar to the previous case, we assume a sequence $x_1, \ldots, x_n$, and a word $x_{\bar{j}}$ is perturbed by Δx. For the vanilla RNN model, the embeddings are sequentially computed as $z_i = \sigma(Ax_i + Bz_{i-1})$. If $x_{\bar{j}}$ is perturbed, then all the $\{z_i\}_{i=\bar{j}}^n$ will be altered. Thus, the attacker can more easily influence all the embeddings.

5.1.2 Illustration of the Proposed Theory

As an illustration of the proposed theory, we plot a comparison of the degree of embeddings variation from two models after changing one word in Fig. 5.1. We observe that, for self-attentive models, the distribution of change on embeddings is sparse after going through the first self-attention layer (layer 1) and then gradually propagate to the whole sequence when passing through more layers. In contrast, the embeddings from LSTM exhibit a denser pattern. To

DOI:10.6814/NCCU202001426



(a) LSTM

(b) BERT

Figure 5.1: L2 norm of embedding variations within (a) LSTM and (b) BERT when one of the input words is swapped, as indicated by the red box.

further validate our analysis, we calculate the ratio of the L2 norms of embeddings variation. Specifically, let $\boldsymbol{z}$ and $\boldsymbol{z}_{\text{adv}}$ denote the embeddings of the original sentence and adversarial input, respectively. We represent relative embedding variation $\mathcal{R}_e = \|\boldsymbol{z} - \boldsymbol{z}_{\text{adv}}\|/\|\boldsymbol{z}\|$. For the GS-EC attack in the sentiment analysis task, embeddings from the LSTM model has an average $\mathcal{R}_e$ of 0.83 whereas for the BERT model it is 0.56 under the same attack by changing one word. This supports our claim that the impact of an adversarial example is more severe on the LSTM model than BERT, which presumably plays an important role in the robustness of self-attentive models.

DOI:10.6814/NCCU202001426

Chapter 6

Conclusions

Throughout this dissertation, we have presented multiple applications of neural networks on various NLP tasks, including classification, sequence generation, and adversarial attacks. This final chapter contains recapitulation of the approaches, summarization of our experimental results, and some directions for future research.

6.1 Theoretical Implications

We demonstrate through extensive experiments that neural networks, including recurrent and self-attentive, can achieve outstanding performances on a wide variety of NLP tasks. More specifically, sequence classification tasks such as detecting the protein interactions mentioned in biomedical literature, or the sentiment in short text can be tackled by RNNs and BERT models. Similarly, sequence-to-sequence learning, which includes translation, summarization, sequence tagging, etc., can also be modeled by these neural networks. In addtion, our experiments indicate that self-attentive models, *i.e.*, BERT or Transformers, are more robust to adversarial attacks than RNNs under small input perturbations.

DOI:10.6814/NCCU202001426

Next, we provide theoretical explanations regarding why the self-attention structure leads to better robustness, as well as illustrative examples that visualize the model's internal variations. We also attempt to generate natural adversarial inputs using the embedding distance as a constraint. Exemplary cases verify that the contextual embeddings from BERT can be used as a measure for semantic similarity.

6.2 Unsolved Problems

Recent studies find that BERT has some knowledge for semantic roles, entity types, and relations [61]. However, there remain a wide array of unresolved issues. As mentioned by Devlin [17], apart from the requirement of large amount of training data and hardware, the learning of linguistic knowledge is not straightforward. Information such as syntax, semantics, pragmatics, and co-reference cannot be intuitively fed into the current model. Structural knowledge such as relation graphs or taxonomy is another struggle for these neural models to try to integrate. Moreover, how do we as humans understand the inference process of neural networks in order to extract useful insights, a question that is worthy of our attention. And last but not least, seeing the wide deployment of BERT-related models in commercial systems, developing a adversarial training scheme as well as devising a more robust architecture based on our findings are equally crucial. The current dissertation serves as a steppingstone for further work along these directions.

DOI:10.6814/NCCU202001426

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[2] Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):S2, 2008.

[3] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/D18-1316.

DOI:10.6814/NCCU202001426

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

[5] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ8vJebC-.

[6] Razvan Bunescu, Ruifang Ge, Rohit J Kate, Edward M Marcotte, Raymond J Mooney, Arun K Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial intelligence in medicine*, 33(2): 139–155, 2005.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[8] R Caruana. Multitask learning. autonomous agents and multi-agent systems. 1998.

[9] Samuel WK Chan and Mickey WC Chong. An agent-based approach to Chinese word segmentation. In *IJCNLP*, pages 112–114, 2008.

[10] Y. C. Chang, C. H. Chu, Y. C. Su, C. C. Chen, and W. L. Hsu. PIPE: a protein-protein interaction passage extraction module for BioCreative challenge. *Database (Oxford)*, 2016, 2016. ISSN 1758-0463. doi: 10.1093/database/baw101.

[11] Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. Unigram language model for Chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Association for Computational Linguistics Jeju Island, Korea, 2005.

DOI:10.6814/NCCU202001426

[12] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[13] Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. Distraction-based neural networks for modeling document. In *IJCAI*, volume 16, pages 2754–2760, 2016.

[14] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128, 2018. URL http://arxiv.org/abs/1803.01128.

[15] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[16] 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*, pages 160–167, 2008.

[17] Jacob Devlin. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Seminar at Stanford*, 2019.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

DOI:10.6814/NCCU202001426

Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[19] Clevert Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 6, 2016.

[20] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 31–36, 2018.

[21] Jeffrey L Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

[22] Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4): 531–574, 2005.

[23] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

[24] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[26] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks

DOI:10.6814/NCCU202001426

for improved phoneme classification and recognition. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 753–753, 2005.

[27] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, PP(99):1–11, 2017.

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

[29] Martin Haspelmath. Word classes/parts of speech. In *International Encyclopedia of the Social and Behavioral Sciences*, pages 16538–16545. Pergamon Pr., 2001.

[30] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[32] Yu-Ming Hsieh, Ming-Hong Bai, Jason S Chang, and Keh-Jiann Chen. Improving PCFG chinese parsing with context-dependent probability re-estimation. *CLP 2012*, page 216, 2012.

[33] Baotian Hu, Qingcai Chen, and Fangze Zhu. LCSTS: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, 2015.

[34] Lei Hua and Chanqin Quan. A shortest dependency path based convolutional neural

DOI:10.6814/NCCU202001426

network for protein-protein relation extraction. *BioMed Research International*, 2016, 2016.

[35] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1875–1885, 2018.

[36] Aaron J Jacobs and Yuk Wah Wong. Maximum entropy word segmentation of Chinese text. In *COLING* ACL 2006*, page 185, 2006.

[37] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

[38] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[39] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[40] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5. URL http://www.sciencedirect.com/science/article/pii/S0893608005801315.

DOI:10.6814/NCCU202001426

[41] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[42] Lishuang Li, Rui Guo, Zhenchao Jiang, and Degen Huang. An approach to improve kernel-based protein-protein interaction extraction by learning from large-scale network data. *Methods*, 83:44 – 50, 2015. ISSN 1046-2023.

[43] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

[44] Chin-Yew Lin. Rouge: Recall-oriented understudy for gisting evaluation. In *Proceedings of the Workshop on Text Summarization*, 2003.

[45] Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 1612164, pages 448–455, 2005.

[46] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[47] Wei-Yun Ma and Keh-Jiann Chen. Introduction to ckip Chinese word segmentation system for the first international Chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 168–171. Association for Computational Linguistics, 2003.

[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv: 1706.06083*, 2017.

DOI:10.6814/NCCU202001426

[49] Xinnian Mao, Yuan Dong, Saike He, Sencheng Bao, and Haila Wang. Chinese word segmentation and named entity recognition based on conditional random fields. In *IJCNLP*, pages 90–93, 2008.

[50] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[51] Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. Protein–protein interaction extraction by leveraging multiple kernels and parsers. *International journal of medical informatics*, 78(12):e39–e46, 2009.

[52] Hirotada Mori. From the sequence to cell modeling: comprehensive functional genomics in escherichia coli. *BMB Reports*, 37(1):83–92, 2004.

[53] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 49–54. IEEE, 2016.

[54] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[55] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

DOI:10.6814/NCCU202001426

[56] Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, page 562. Association for Computational Linguistics, 2004.

[57] Yifan Peng and Zhiyong Lu. Deep learning for extracting protein-protein interactions from biomedical literature. In *Proceedings of the 2017 Workshop on Biomedical Natural Language Processing*, 2017. to appear.

[58] Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50, 2007.

[59] L. H. Qian and G. D. Zhou. Tree kernel-based protein–protein interaction extraction from biomedical literature. *Journal of Biomedical Informatics*, 45(3):535 – 543, 2012. ISSN 1532-0464.

[60] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Preprint*, 2018.

[61] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*, 2020.

[62] Sebastian Ruder. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

[63] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

DOI:10.6814/NCCU202001426

[64] BRUCE W SUTER. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):291, 1990.

[65] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[66] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[67] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272, 2018.

[68] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[69] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX (236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[71] Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. Chinese word segmentation with maximum entropy and n-gram language model. In *COLING* ACL 2006*, page 138, 2006.

DOI:10.6814/NCCU202001426

[72] Warren Weaver. Translation. *Machine translation of languages*, 14:15–23, 1955.

[73] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(88)90007-X. URL http://www.sciencedirect.com/science/article/pii/089360808890007X.

[74] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[75] Chia-Wei Wu, Shyh-Yi Jan, Richard Tzong-Han Tsai, and Wen-Lian Hsu. On using ensemble methods for Chinese named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 142–145, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-0122.

[76] Jin Yang, Jean Senellart, and Remi Zajac. Systran's Chinese word segmentation. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 180–183. Association for Computational Linguistics, 2003.

[77] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *arXiv preprint arXiv:1805.12316*, 2018.

[78] Xiaofeng Yu, Marine Carpuat, and Dekai Wu. Boosting for Chinese named entity

DOI:10.6814/NCCU202001426

recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 150–153, 2006.

[79] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[80] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc., 2015.

[81] Hai Zhao, Chang-Ning Huang, Mu Li, et al. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117. Sydney: July, 2006.

[82] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1BLjgZCb.

DOI:10.6814/NCCU202001426