Imperial College London

Department of Computing

Attention-Based Sentiment Analysis on Social Media

Author:
Zheyu Ye

Supervisor:
Dr. Anandha Gopalan

Submitted in partial fulfillment of the requirements for the MSc degree in Master of Advanced Computing of Imperial College London

September 9, 2019

Abstract

The application of sentiment analysis in social media has attracted much attention. To date, the focus of research has turned to the interpretation of textual information. Several mature models are proposed in natural language processing and opinion mining, combined with grammar-based linguistic methods and statistical-based machine learning methods. The mode of early pre-training with the following transfer learning has been proved to be suitable for most downstream tasks. In this thesis, we propose several models that based on BERT trained-model and utilize a Convolution Neural Network and attention mechanism to capture textual correlation. For graphical symbols (e.g. Emoji and Emoticon), which are commonly used in social media, we applied contextual attention to interpret the interaction between Emoji and Text. Additionally, we simulate the probability distribution of Emoji in practical application and construct a dataset to feed Emoji. The expressiveness of Emoji embedding has been enhanced to avoid imbalance in competition with text. To better understand the actual use of tweet, we build a page-friendly web application, which collects manual annotation data and provides visitors with a reliable API for sentiment analysis of tweet.

Acknowledgements

It seems that I am reaching the end of the journey of a student, Imperial College London is undoubtedly the most impressive stops, giving me a broad academic platform with abundant research resources and a distinctive horizon.

This master thesis was completed under the supervision and careful guidance of my supervisor – Dr. Gopalan Anandha. From the project selection to the final delivery, his serious scientific attitude, rigorous academic spirit and perfect working style let me have a deep understanding and clear direction.

Last but by not least, for everyone in the Department of Computing , I would like to thank all the teaching fellows, senior, intimate friends, who I have met last year. Of course, I appreciate the endless support, and the care given by my parents is always my strong backing.

Thanks for all your encouragement!

Contents

Chapter 1

Introduction

1.1 Overview

As the development of communication technology has enhanced, our life is digitised and informational. At present, instant messaging is a reality, not a distant dream, which made social media, such as Twitter[1], Facebook[2], are full of massive Internet users and data. Nowadays, instant messaging provides a new platform for social needs.

It is a common phenomenon that people use social media to vent emotions and express opinions by photos, Emoji, text, articles, microblogging, and so on. In a previous work in 2008 [43], "What other people think" is always an everlasting topic of information analysis and decision-making. Conveying a wealth of information and expressing diverse emotional content, social media such as twitter has sparked our interest. Technology extracts these rich and diverse contents and maps them into interesting tasks, reflecting the original intention of this paper, that is, natural language processing and emotional analysis of tweets. Considering that language is the most complex signal that a machine can understand, the information contained in a tweet may be difficult to capture and model.

Since the 1930s, Natural Language Processing (usually abbreviated as NLP) technology has been regarded as the core part of the field of Internet information process-

[1]https://twitter.com/
[2]https://www.facebook.com/

ing, aiming at automatically interpreting and expressing human languages, which requires cooperation in other fields such as linguistics. Psychology, philosophy, cognitive science and machine learning based on probability and statistics [24, 22, 61]. Through several mature NLP frameworks, such as Named-Entity Recognition (NER), Semantic Role Labelling (SRL), Part-Of-Speech Tagging (POS), and some deep learning frameworks for complex neural networks, NLP enables computers to perform a wide range of tasks related to natural language, such as information extraction, question answering, sentiment analysis, and machine translation. [7].

Sentiment analysis, as a core part of Natural Language Processing, is the process of conjecturing and classifying opinions expressed in a piece of text [43, 47]. Different from traditional text mining, sentiment analysis focuses on attitudes instead of facts [55]. Thus, the application can be used on a variety of tasks, for example, analysis of personal microblogging such as Twitter, evaluation of the positivity or negativity of news events from New sites (such as BBC[3], CNN[4], Guardian[5], etc.).

Considering that the existing text classification and sentiment analysis tasks are relatively mature, the following objectives for further expansion are listed as below:

- Model the language environment of social media and perform sentiment analysis on input tweet.
- Construct a dataset from human annotators through a self-built web application, and use automatic annotation approach to enrich the sample size and diversity.
- Compare lexicon and rule-based methods and statistics-based methods, and test them on multiple dataset including own collected dataset.
- Apply a Autoencoder based per-training model with various classifies, and strengthen the information capture ability based on attention mechanism.
- Combine with Emoji and emoction, proposed an Attention-based classification model, and explore the impact of the text and Emoji on the overall sentiment, verify the prior probability of Emoji

[3]https://www.bbc.co.uk/
[4]https://edition.cnn.com/
[5]https://www.theguardian.com/uk

1.2 Challenges

Compared with other tasks such as image processing, natural language processing has a long-term accumulation of rules, which are constantly updated and present abstract expression features (such as singular/plural, pronouns, words, etc.). Language is an ambiguous and subjective expression, which conveys all kinds of information to everyone and the background environment [43]. The consensus that language comprehension plays a key role in communication is undoubtedly a major obstacle to machines for the following reasons.

The first dilemma of emotional analysis is that when we read some texts, language users can understand the connotation (intentional meaning) from the denotation (literal meaning). For example, how can anyone stay in a movie, which expresses negative opinions [42, 55]. Mining implicature, irony and irony through expression requires machines with reasoning or sense of humour, like humans, which transcend basic grammatical rules and lexical analysis [28, 55].

Another bottleneck of sentiment analysis is the problem of using the same corpus to work with different topics or fields (i.e. huge discrepancy of language style between official press releases and personal tweets) [28]. A sentiment-known word from a topic is probably invalid in others [55], still less slang and highly innovative word in Twitter [23]. This phenomenon can be attributed to the lack of the ability about self-learning and migration of language learning which allow humans to know the meaning of a word that had never been seen before, solely depending on context and root affixes [48].

Emotional analysis in social media is facing a new challenge as the Internet world becomes more diverse. For Twitter, it could increase the contextual sparsity in 280 characters (doubling from 140 characters on November 7, 2017), and now it's full of fresh words, online slang and Emojis [6] , leading to the cruel training environment for practical algorithm. At the same time, emotional symbols are widely respected on Twitter, bringing benefit to the expression of diversity (e.g. using ‘:-) ’, ‘:)’, ‘=)’, ‘:D’, etc. for expressing positive attitude while negative by ‘:-(’, ‘:(’, ‘=(;, ‘;(’, etc.) [23, 41] along with challenges to be confronted with.

[6] a commonly used electronic expression, such as hieroglyphs

1.3 Contribution

The dominant contributions in this Master thesis can concluded as below:

- Twitter API is used to crawl raw tweet and obtained an adequate dataset with automatic and manual annotation after data cleaning with contraction mapping and graphic characters translation.
- A beautiful and powerful web application is constructed to provide a manual annotation platform that allows visitor to analyze tweet sentiment of searched tweets or twitter users.
- BERT pre-training model is combined with multiple classifications (concatenated CNN and linear), which can be applied to pure text tweets classification. The expression ability of the model on multiple datasets is verified, and the weighted sampling, weighted cross-entropy and focal loss are compared on unbalanced data.
- Based on a well-trained multi-head self-attention Transformer encoder, an Emoji attended model is proposed focusing on the contextual semantics between text and Emoji. The performance of the model is compared with that of the pure text model with multiple metrics considering Emoji as additional information.
- A dataset is constructed to simulate the real scenario of Emoji usage in social media. The Emoji embedding is gradually strengthened under the moistening of the dataset and can be used in a subsequent experiment.

Chapter 2

Preliminaries

In this chapter, we will provide some prerequisite knowledge of Natural Language Processing and Sentiment Analysis as well as the main approaches used currently. Starting with introducing a variety of ways that how computers represent the words and sentences, we will give a brief review of Deep Learning, including its basic architectures such as CNN, RNN and the prevailing attention mechanism and Transformer.

2.1 Foundation

For the task of sentiment analysis, it can be divided into the following types: document level, sentence level, and aspect level (feature-based) [28]. Given a set of opinionated documents (e.g. a movie review or a product review), the document level determines the sentiment polarities (also known as semantic orientations) [48]. The sentence level might be more breakable to be analyzed. While for the compound sentences, it may express different opinions on different aspects, which is a severe problem for making a global assessment. For aspect level, the task is to analyze the sentimental tendency of the sentence on a given aspect (as known as features, e.g. "camera" or "chip" in a product review of a smartphone).

Two main ideas, including lexicon and rule-based methods and statistics-based methods, were proposed and applied to the task of Sentiment Analysis. The former one, as its name suggests, uses the language laws and structures proposed by professional linguists and constructs a model with universal applicability. As known as Machine

Learning, statistics-based method is trying to express rich emotional information and present a complete mental journey, in order to rid of the stereotype of cold and ruthless.

2.1.1 Lexicon and Rule-Based Methods

For all levels of lexicon-based approach, the most elemental but essential lexicon and rule-based method is evaluating the semantic orientation of a single word token (could be a word, an Emoji, an emoticon, even a punctuation mark). Resorting to a sentiment lexicon, established manually or automatically [23, 48, 17], does help machine to conjecture the word polarity. There are several available public lexical dictionaries that can collect opinion words and polarity, such as Lexical Dictionaries:LIWC [1], Multi-Perspective Question Answering (MPQA) [2] [51], Affective Norms for English Words (ANEW), the General Inquirer (GI), and SentiWordNet [3].

For the task of mining word polarity to interpret contextual polarity or describe objects, more efforts need to put into analysing syntactic and semantic meaning that based on other sub-tasks of Natural Language Processing, such as Named-Entity Recognition (NER), Semantic Role Labelling(SRL) and Part-Of-Speech Tagging (POS). Furthermore, as Pak and Paroubek reported in *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*, POS-tagger was used on the Twitter corpus and obtained the same distributions of sentence elements on positive and negative set [41].

2.1.2 Statistics-Based Methods

Statistics-based methods involves training a classifier from labelled data [42], and some basic supervised learning algorithms were engaged in sentiment analysis, such as Naive Bayes [37], maximum entropy (MaxEnt) [36], and Support Vector Machines (SVM) [35, 13].

At the cutting edge in the deep learning as a machine learning branch, Deep Learn-

[1]http://liwc.wpengine.com/
[2]https://mpqa.cs.pitt.edu/
[3]http://swn.isti.cnr.it/

ing is aimed to discover the rules of latent feature and generate the corresponding extends development pattern. Currently, Deep Learning is widely used in NLP, because almost every NLP task can be formulated as a classification task. As Collobert et al. described, the purpose of NLP task is to extract a rich set of hand-designed features from the sentence and use it as the input to the classification model [7].

However, with the more complex corpus and more differentiated application scenarios, traditional machine learning methods are gradually unable to adapt to the high standards of modern NLP missions. On the other hand, compelling and adaptable approaches to represent word embedding were proposed such as Word2Vec and GloVe, and deep learning and neural networks come back in sight of researchers and practitioners. As an indispensable part of this ongoing project, neural networks, including CNNs and RNNs, is advantageous for excavating non-linear correlation and abstract features from the magnanimous data.

2.2 Word Embedding

In order to let the machine capture the human thought, the first step is how to represent words as data, which is crucial for all NLP tasks. Word Vector is a sound solution that encodes word tokens into vectors whilst providing a way to distinguish similarity and difference between digitised words [7].

Embedding representation architectures conserve the linear regularities of inter-words [32, 45]. Here is a canonical syntactic analogy that shows how the word vector performs self-addition and subtraction, as mathematical operators, to show the correlation among words [33].

“king - man + woman = queen”

Studies show that well-trained word vector models can find not only the syntactic relationship between words and words but also the correlation information. The representation of embedded words can be fine-tuned according to the different input of different downstream tasks. This application scenario can generate many elegant pre-training models.

One-hot Vector

Traditional method of the word representation, such as one-hot vector, encodes a word into a poly-dimensional vector. Each specified word among the vector has its unique index as the following:

$$\boldsymbol{w}^{\text{Imperial}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{College} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{London} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{Computing} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

It can be expected that this method usually generates a vector with enormous dimensions to cover the complete dictionary, and loses the ability to capture syntactic and semantic relationships. An intuitive idea to fix this issue is to let the synonyms have similar vectors. Recently, some advanced methods as remedies were introduced, which were engaged in training massive data through neural networks.

Word2Vec

An iteration based method, Word2Vec[4], was created by Google in 2013 [32]. As a shallow Window-Based Methods, the core idea of word2Vec is to predict the relationship between each word and the word's context (within the window). With adjustable window size (the number of surrounding words), Word2Vec architectures have achieved good results under multiple sub-tasks of NLP

Two constructive models were proposed, including continuous bag-of-word (CBOW) and skip-gram (SG) models. CBOW predicts the centre word according to the context (the surrounding words); whereas, the SG model does the reverse logic work that predicts the context words given a centre word. The structure of these two models are shown in Figure 2.1.

Speaking of the derivation of word2vec, we have to point out the and N-grams model. Bag-of-words is a simplified representation that disregards grammatical rules and contextual meaning through only analysing the quantity statistics [16].

[4] https://code.google.com/p/word2vec/

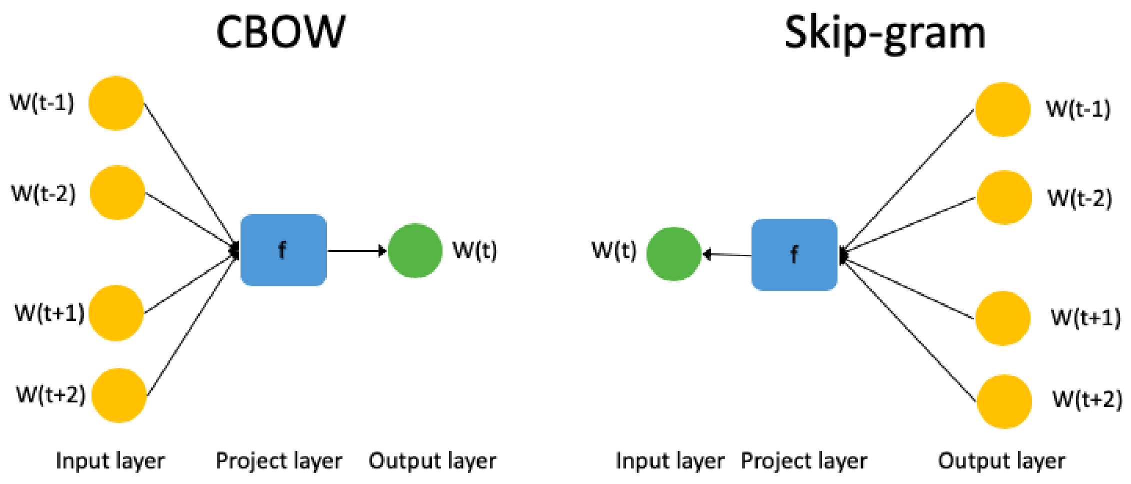


Figure 2.1: Two designed model architectures CBOW and Skip-gram

N-grams, as a window-based method, is a language model for calculating the probability of a sentence, that is, the probability of judging whether a sentence is reasonable [31]. Based on the Markov assumption [11] that the appearance of the N-th word is only related to the previous N-1 words, N-grams compute the product of the likelihood of occurrence of each word to get the probability of a whole sentence. N-grams model is heavily used in search tools (e.g. Google), predicting the next word will be entered. Combining the design concepts of these two models, CBOw and SG model use the contextual dependencies to update the transfer matrix by gradient descent continuously, and yield the word representation.

The contribution of this far-reaching paper to the field of Natural Language Processing goes far beyond that. Two moderately effective optimisation designs – Hierarchical softmax and Negative sampling also have profound enlightenment and reference meaning. Inspired by Hierarchical [34] and Huffman Tree [21], Hierarchical softmax uses the words frequency derived from statistics to create a Huffman tree with depth of $\log V$ (V presents the size of vocabulary). The coupling degree of word-to-word is increased with the reduction of the target probability's computational complexity from the initial O(V) to O($\log V$). Negative sampling allows a training sample to update only a small portion of the weight, whereas the traditional method is to update all weights for each training sample. Those mismatched results in output layer (called negative words) are randomly selected in a small proportion to update the corresponding weight, which will reduce the amount of calculation in the gradient descent process.

GloVe

In 2014 another important word embedding method was put forward named GloVe[5] (Global Vectors for Word Representation) [45]. It averages the global occurrence statistics from dataset, and learns word representations efficiently with the help of squared error method. This method can effectively deal with the polysemous in the English, realising a decent division of multi-meaning words.

From an intention that an interpretable word embedding should be related to the ratio of word co-occurrence probabilities, not their probabilities themselves, Glove considered both macrocosmic information and microcosmic information, using window-based co-occurrence matrix, making up the dependencies between in-window and out-door window, which is what word2Vec model lacks.

2.3 Convolutional Neural Network (CNN)

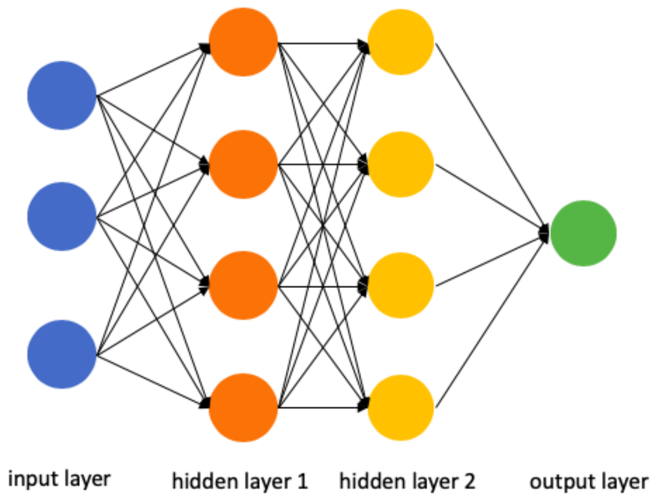
Convolutional Neural Network (CNN) is a Deep Learning algorithm which is composed of multiple layers of different functions (i.e. convolution, pooling, activation layer). Mimicking the architecture of the human brain, neurons are connected with others, receiving the weighted sum of its inputs and passing it through an activation function. The network can be construed deeper progressively with the continuous alternation of input and output [14, 40]. As shown in (Figure 2.2).

In the field of computer vision, Convolutional Neural Network has been reputed to be a revolutionary tool with vast prospective. Besides that, it has been proven that it is also very suitable for time series modelling of words after being introduced into NLP [6].

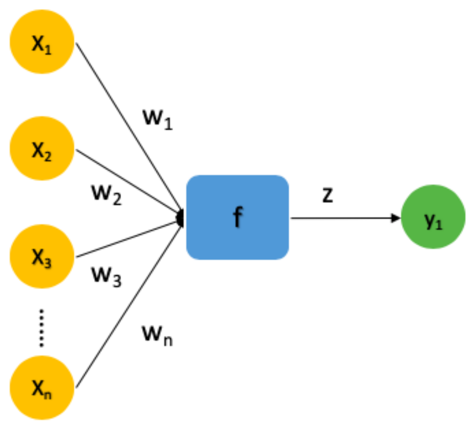
Kim, Yoon offered an alternative solution for sentence-level classification called TextCNN that captures sentence information through convolutional operations. The multidimensional vector generated by convolution is mapped into an output probability for each category, in which the process of convolution can be reflected by the following equations:

$$\boldsymbol{w}_{1:n} = \mathbf{w}_1 \oplus \boldsymbol{w}_2 \oplus \ldots \oplus \boldsymbol{w}_n \tag{2.1}$$

[5]http://nlp.stanford.edu/projects/glove/



(a) Four layers fully connected CNN



(b) Architecture of Neurons

Figure 2.2: A simple fully connected CNN structure

where $\oplus$ represents the concatenation operator,$\boldsymbol{w}_i \in \mathbb{R}^d$ is a d-dimensional word vector located in i-th of the input sentence (padded where needed). Apply an convolution operation on a sub-matrix $\boldsymbol{w}_{i:i+j}$ (row i to row $i+j$), we can get a feature map $\boldsymbol{c} \in \mathbb{R}^{n-h+1}$ as:

$$c_i = f\left(\boldsymbol{w}_{i:i+h-1} \cdot \boldsymbol{k}^T + b\right) \tag{2.2}$$

Where h is the size of window, $\boldsymbol{k} \in \mathbb{R}^{h \times d}$ is the filter, and f is non-linear activation function with bias b. By concatenating convolution kernels of different window sizes, the receptive field range and accuracy can be strengthened.

2.4 Recurrent Neural Network (RNN)

labelsec:lstm Recurrent neural networks are more suitable variant of neural networks for Natural Language Processing with the excellent ability to process sequential data. RNNs are usually equipped with a memory unit to store the correlation with previous input that ensures the integrity of the context in sentences.

The below equation interprets the concept of a simple RNN with hyperbolic tangent function as activation function

$$h_t = \tanh\left(\left(\boldsymbol{A}\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)\right) \tag{2.3}$$

where h_t is the current state at this time node in sequence as h_{t-1} is the previous one and x_t denotes the current input. Matrix $\boldsymbol{A}$ is the the parameter of transform function without considering basis. Figure 2.3 illustrates an unrolled RNN that reuses the same weight matrix at every time step. Under this architecture, a deep network can be constructed without considering the length of the input, while Bidirectional RNN [50] connects opposite directions and generate a shared output, which is widely used in live application such as speech processing [14, 40]. Based on this sequence structure, many variants are spawned such as Long short-term memory (LSTM) [20], Gated Recurrent Unit (GRU) [5].

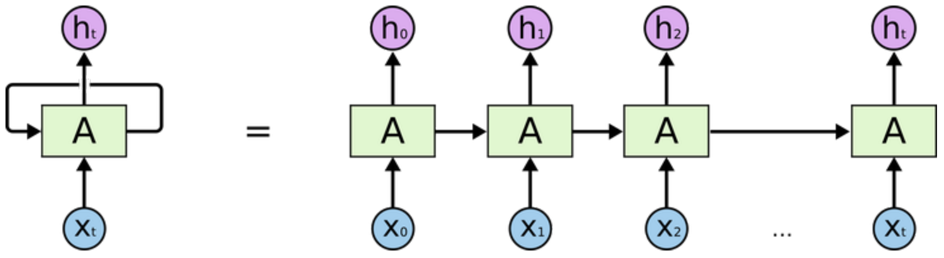


Figure 2.3: Illustration of an unfold RNN (adapted from Olah's blog [39])

Based on RNN, Long Short-Term Memory (LSTM) networks [20] is useful for reserves values over arbitrary time by regularly forgetting irrelevant information through three introduced three gates, including input, output and forget gate.

$$
\begin{gathered}
\dot{f}_t=\sigma\left(W_f \cdot \boldsymbol{X}+b_f\right) \\
o_t=\sigma\left(W_o \cdot \boldsymbol{X}+b_o\right) \\
o_t=\sigma\left(W_o . \boldsymbol{X}+b_o\right) \\
c_t=f_t \odot c_{t-1}+i_t \odot \tanh \left(W_c . \boldsymbol{X}+b_c\right) \\
h_t=o_t \odot \tanh \left(c_t\right)
\end{gathered} \tag{2.4}
$$

It can be observed from above Equation 2.4, at time t the current input x_t and the past hidden state h_{t-1} combined as a new input $\boldsymbol{X} = [h_{t-1}, x_t]$. f_t, i_t, o_t represent the gate of forget,input and output while c_t, called cell state, is an extra helping regulate information. $\odot$ refers to the Hadamard product to performs element-wise multiplication. σ denotes the sigmoid function mimicking the status of the gate of open or close.

A highly engineered and slightly changed architecture is the Gated Recurrent Unit (GRU) [5, 40], keeping the function of controlling date flow with only two gates including update gate and reset gate.

2.5 Encoder-Decoder

2.5.1 Seq2seq

Under the encoder-decoder structure, sequence-to-sequence (seq2seq) model receives and outputs the serialised signal [53, 5]. The Encoder converts a variable-length signal sequence into a fixed-length vector representation (latent vector c), while decoder turns this fixed-length vector into a variable-length output sequence as igure 2.4 showcased.

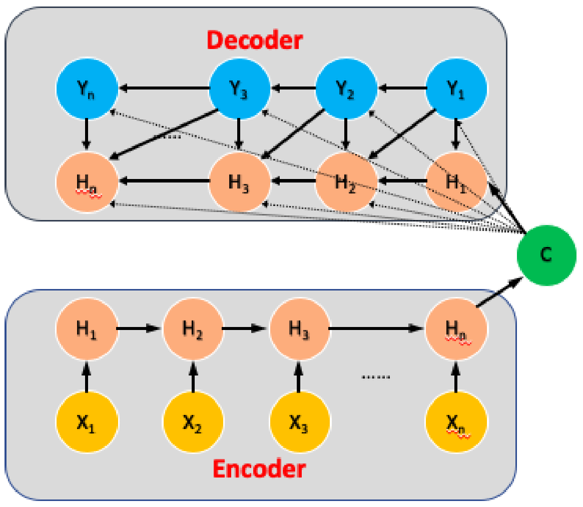


Figure 2.4: An illustration of the seq2qeq model

This RNN-based model solves a mapping problem involving variable-length inputs of $x$and variable-length outputs of $y$instead of fixed inputs and outputs. However, due to the length limitation of c, it is often impossible to fully represent the information of the entire sequence. Besides that, under such a mechanism, the information carried in advance hidden state can be slowly diluted in the decoding process, resulting in an inaccurate result.

2.5.2 Attention Mechanism

The inadequacy of the encoder-decoder model has led to the proposal of attention mechanism, which was firstly applied in Neural Machine Translation (NMT) [2] with an amazing performance.

When people watch a picture or read a piece of text, they can often grasp the highlight without tips, which is the overall awareness pursued by all deep learning models nowadays. Much as human observation mechanism, the Attention Mechanism teaches machines an ability to observe one specific part of the scene as needed, rather than looking at the scene from the beginning to the end. At the same time, Attention Mechanism will memorise positions and learn regularity of the occurrence of highlights in the scene, so that ”attention” can be focused on this part in the next similar scenes. By allocating weighted attention on the input source, machine comprehension has been promoted both in Computer Vision and Natural Language Learning(i.e. Sentiment Analysis [57] and Image Caption [59], etc.)

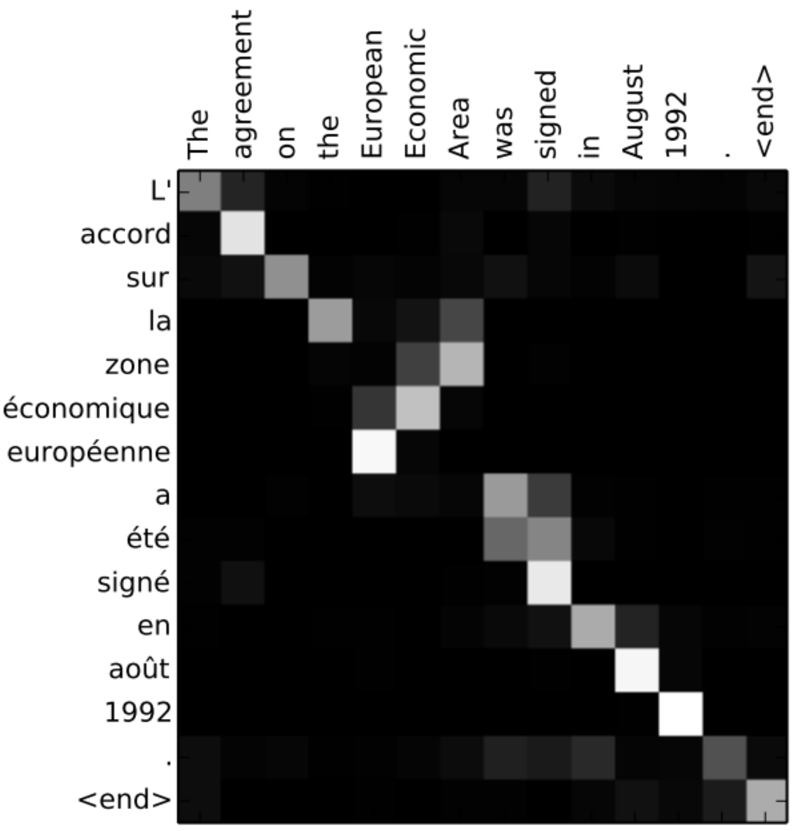


Figure 2.5: Attention visualisation of a French-English translation [2]

Attention Mechanism supplies a remedy to traditional seq2seq model by allowing the decoder to pay ”attention” on apart parts of the input and find focus [2]. In an example of English-French translation in Figure 2.5, the brightness shows the attention of word aliment. For instance, English words “European Economic Area” shows higher weight on output “zone économique européenne” in attention matrix, which is the corresponding French translation, reflecting many-to-many text alignment.

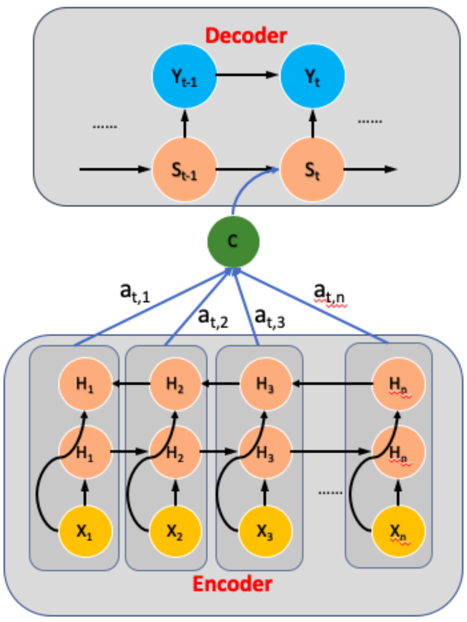


Figure 2.6: The graphical illustration of the attention-based model

As Figure 2.6 shown, attention-based model uses a bidirectional RNN as an Encoder and defines a conditional probability of output y in decoder as

$$p\left(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}\right) = g\left(y_{i-1}, s_i, c_i\right) \tag{2.5}$$

where s_i is an RNN hidden state of decoder for time i, computed by

$$s_i = f\left(s_{i-1}, y_{i-1}, c_i\right)$$

Focusing attention on the i-th word of the input sequence and its neighbour, each annotation h_i represents the information about the whole input sequence. Different from traditional seq2seq model of direct accumulation calculation, the context vector c_i is computed as a weighted sum of a sequence annotations h_i:

$$c_i = \sum_{j=1}^{T_n} \alpha_{ij} h_j \tag{2.6}$$

the corresponding weight α_{ij} indicates the extent to which the jth word at the source affects the i-th word at the target end, which is a softmax result of alignment model e_{ij}:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \quad e_{ij} = a\left(s_i, h_j\right) \tag{2.7}$$

There are many different ways to calculate the alignment model e_{ij}, representing different attention models. Vividly understood the alignment model as scoring the level of attention, alignment model can be also called sore function, and the simplest and most commonly used sore function is the dot product matrix. To be specific, that is a matrix Multiplication for calculating the current hidden state s_t of the target end (decoder) and the current hidden state h_s of the source end (encoder), such that $\boldsymbol{e}_t = \left[s_t^T \boldsymbol{h}_1 \cdots s_t^T \boldsymbol{h}_N\right]$, where hidden states of the encoder $\boldsymbol{h}_1 \cdots \boldsymbol{h}_N$ can be called as keys in attention mechanism, and the hidden states of current decoder denoted as query (describe details in section 2.6 Transformer).

2.6 Transformer

2.6.1 Integral Structure

In the paper of "Attention Is All You Need", [56] devised a network architecture, the Transformer, casting away the old-fashioned CNN and RNN that might raise the issue of remaining long-distance dependencies, even though gained ground already on all fronts in this century. While the Transformer is a model single-handedly based on attention mechanisms.

In the model proposed in the original paper (see in Figure 2.7), both the encoder and the decoder are composed of 6 layers of repeating structures. Multi-head self-attention (an ensemble with a number of attention units) and position-wise feed-forward network are both applied in the encoder and decoder, while decoder has another multi-head context-attention mechanism inside. Much as seq2seq model, context-attention mimics the same function in the junction of two coders, learning to attend the interesting things in linguistic structure. Besides that, Relative information and absolute position correlation are injected with positional since positional information is ignored by attention mechanism. [56].

2.6.2 Residual Neural Network

In the structure shown in Figure 2.7, an Add & Norm layer is attached to each major functional layer. In practical terms, "Add" represents residual connection, which is

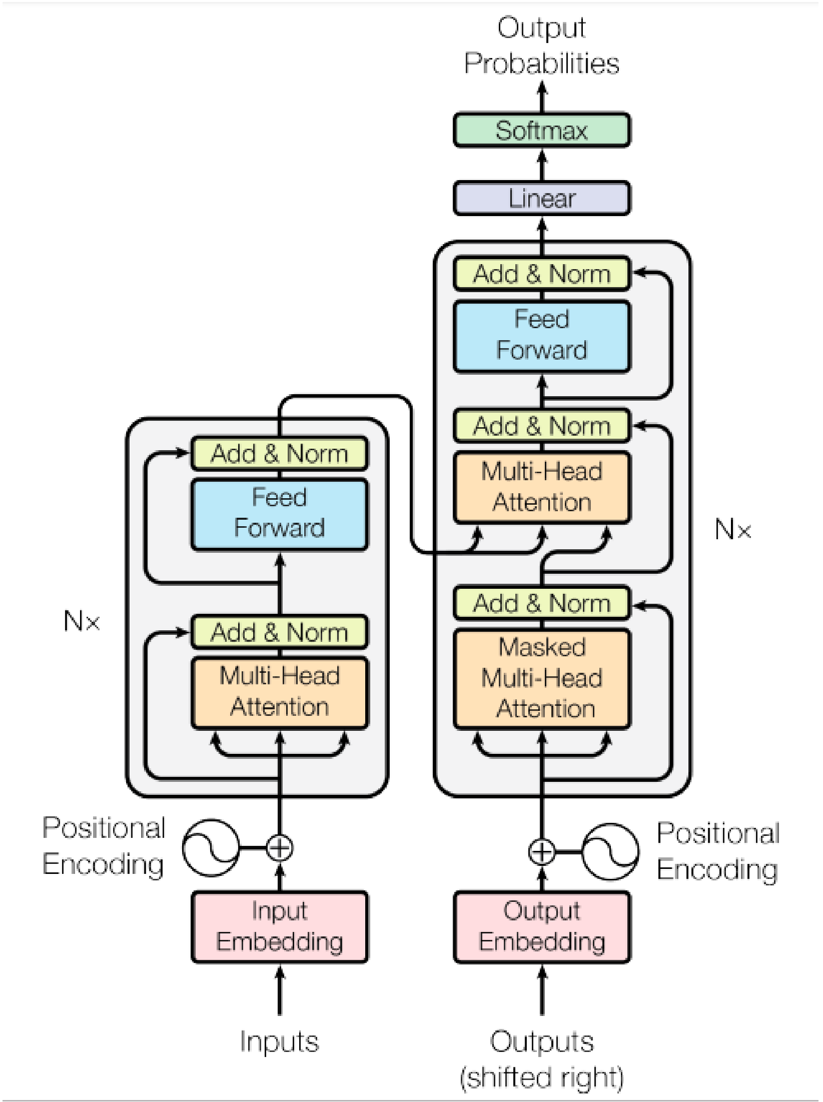


Figure 2.7: The model architecture of the Transformer [56]

to solve the problem of difficult training in deep neural networks. By identically transmitting the information of the previous layer to the next layer, it can effectively focus only on the residual [18]. “Norm” represents Layer Normalization, which normalizes the activation value of the layer in a same probability distribution [1].

It stands to reason that when we stack a deep model, it is natural to assume that the performance will be better. However, unsatisfactory results could still exist with problems of model degradation, even if we solve over-fitting and gradient explosion or vanishing by normalized initialization and intermediate layers [18]. Since nonlinear activation functions (e.g. ReLU) are often added to the network, the input-to-output process for each layer is almost irreversible. The original input is hard to inferred from the output and might be forgotten in a tortuous training process for a deep network. As seen in Figure 2.8, the shortcut connection works by remembering the original source x in the training process as $H(x) = F(x) + x$ where $H(x)$ denotes the output. This mechanism is quite simple but plays an important role, alleviating information loss and model degradation raised by the deep network structure.

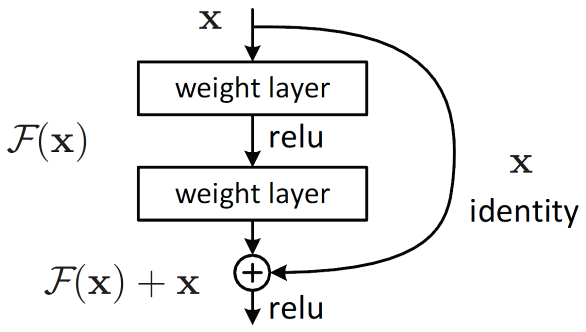


Figure 2.8: The shortcut connection [18]

2.6.3 Self-Attention

The traditional attention mechanism discussed in the previous section 2.5.2 ignores the dependence between words and words in the source or target sentences. In contrast, self Attention can obtain the dependence between the source end and the target end, and it can effectively complement their own information with about the dependence between the source and the target on source end or target end, that is, a single sequence compute their own attention scores. The context-attention is the attention between the encoder and the decoder, which is the difference between two different sequences.

The basic but essential part of the Transformer model is the attention which can be calculated by a block named Scaled Dot-Product Attention through multiple mathematical operations, where the $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ present Query, Key and Value. As shown in Figure 2.9.

Similarly to attention in encoder-decoder model, the purpose of Scaled Dot-Product is to return a weighted score of source sequence and target sequence, which can be computed as:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2.8}$$

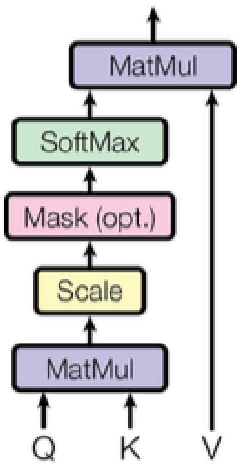


Figure 2.9: Scaled Dot-Product Attention [56]

Vector Query, Key and Value are the results multiplied by word embedding $\boldsymbol{X}$ and parameter matrix to be trained $\boldsymbol{M^Q}, \boldsymbol{M^K}, \boldsymbol{M^V}$ respectively. In this process, queries play a role as locking focus target and keys used to measure dependencies that how close the target word need to query to the current word, and values is a weighted matrix. They seem to be relatively complicated, but all of which do essentially the same thing and are randomly initialized parameters that need to be trained.

As for the "scaled" in the module, it is reflected by $\sqrt{d_k}$ that a scale factor (the square root of the dimension of the Key vectors used in the paper – 64). It is ensured that the probability distribution after softmax does not have a large difference with some peak values while others are nearly vanish, triggering a series problems of small gradient.

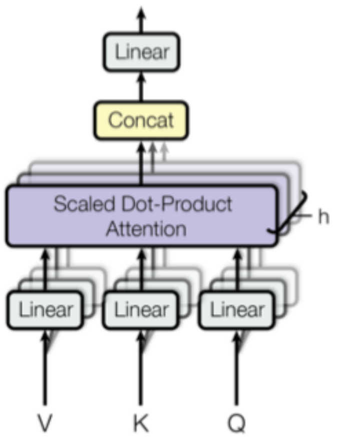


Figure 2.10: MultiHead Attention Mechanism [56]

Multi-head attention mechanism allows the model to have multiple $\boldsymbol{M^Q}, \boldsymbol{M^K}, \boldsymbol{M^V}$ vectors and combines the results of each part. In the specific implementation, perform multiple different initialisations (using 8 heads in the original paper), then linearly map again to get the final output. Unique attention mechanism might lead to focus on local information. However, multi-head attention mechanism makes each position can attend to all other positions in the previous layer, holding contextual information without the limitations of distance [56].

As can be seen in the Figure 2.7, the encoder simply superimposes these essential elements, where the key, query, and value are all from the output of the previous layer. A slight change occurred in the decoder. To be specific, masking mechanism is added to the beginning layer, avoiding getting the prediction of the back, and ensure that the decoder can only attend to the known output processed and placed at the front. As the second layer in the decoder, encoder-decoder attention layer, uses the output of the encoder as $\boldsymbol{K}, \boldsymbol{V}$, ensuring contextual attention can be traversed to each position of the input sequence.

2.6.4 Comparison

Transformer abandons traditional Convolution Neural Networks and Recurrent neural networks, using only attention with full connectivity as the main architecture. Although such an architecture does not escape the routine of traditional deep learning, such a fancy design provides an enlightened idea for follow-up researchers and has also achieved excellent results in various NLP fields.

The Transformer uses the attention mechanism to enable model to directly attend on each part of the sequence signal or other features (like previous hidden states), making no distance between any two words which is very beneficial in solving the thorny problem of long-term dependence in Natural language processing.

As Vaswani et al. discussed in the original paper, compared to CNNs and RNNs, self-attention presents the obvious superiority on time complexity of each layer when the sequence length n is smaller than the representation dimensionality d, which is most often the case. In addition to this advantage, self-attention allows for significantly more parallelization and obtains a more interpretable language model, which is fit with current hardware environments (mainly GPU).

The attention mechanism and Transformer and is now widely used in various models, such as BERT [10], GPT [49] and Transformer-XL [8]. BERT (Bidirectional Encoder Representation from Transformers) is in an atmosphere that allowed it to soar, broken 11 natural records of language processing tasks. It is foreseeable that BERT, as the topmost achievement breakthrough in Natural Language Processing, will bring a world-shaking change to this field, details of which are discussed in section 3.1.3.

Chapter 3

Background

In this chapter, we will review some prior works of NLP and sentiment analysis, with a comparison of multiple pre-trained models of two different training objectives, namely Autoregressive (AR) language model and Autoencoder (AE) model, and conducted an in-depth study of several pre-trained models with outstanding performance, such as GPT, BERT, and XLNet, which is quite different from previously fixed word embedding approach. We will also introduce a non-machine learning combined with linguistic knowledge and syntax rules.

3.1 AR and AE Language Model

The pre-trained model extracts language information according to the self-dependency of the language through an enormous unlabelled corpus without a specific training task generating a reliable word embedding with rich syntactical and semantic information that can be fine-tuned by downstream tasks. The pretraining objectives can be mainly classified as Autoregressive (AR) language modelling and Autoencoder (AE) that both of which has some mature model frameworks.

Autoregressive (AR) language model predicts the next possible word according to the previous content, which is a left-to-right language model, or vice versa [60]. Like GPT [49] and EMLo [46] these Autoregressive language models, the concept of seeking to estimate the probability distribution based on a single direction sequence lead them to generate a gap from real contextual information [60]. However, it is suitable for some downstream NLP tasks, such as generative tasks involving summaries, machine translations, etc. The generation of context is a harmonious process, from left to right, which naturally matches the Autoregressive language model that is a generative model with tractable probability.

The exertion of Autoregressive language modelling methods depends on a reasonable probability density distribution to predict the next signal based on past traversed ones. Compared to Autoregressive (AR) language model, Autoencoder-based (AE) model tends to a destination of reconstructing the original data from fragmentary input [60].

3.1.1 ELMo

In the previous work of word2Vec [32] in 2013 and GloVe [45] in 2014, each word corresponds to a vector, which is shrivel for polysemy. Different from stereotyped word representation, ELMo puts forward an inference based on the surrounding context. One of the obvious benefits of doing this is that for polysemous words, they can be rationally interpreted rather than robotically transfer word meaning as rote. For example, apple can be analysed as the famous technology company or a common fruit, according to the semantic environment.

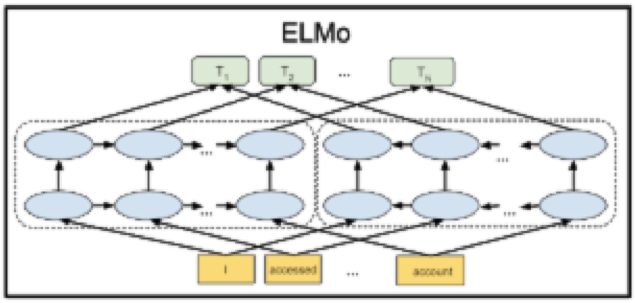


Figure 3.1: Model architecture of ELMo with bidirectional LSTM [10]

Taking LSTM as the core technique (see in Figure 3.1) , ELMo uses a deep bidirectional language model (biLM), to model both complex features of verbal meaning and cross contextual expression. A biLM considers both left-to-right (forward) and right-to-left (backward) directions, training parameters by maximizing the log-likelihood as:

$$\begin{aligned}\textstyle\sum_{k=1}^{N} &(\log p\left(t_k|t_1,\ldots,t_{k-1};\Theta_x,\boldsymbol{\Theta}_{LSTM},\Theta_s\right)\\ &+\log p\left(t_k|t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s\right)\Big)\end{aligned} \tag{3.1}$$

As the above equation shown, the ELMO can be categorised as an Autoregressive language model that concatenates the hidden states of the LSTM in two directions to incarnate the bi-direction. The significance of ELMo exists in Transfer learning that a pre-trained model under unsupervised learning can be used for other downstream NLP tasks after fine-tuning.

3.1.2 GPT

Proposed by OPENAI, GPT (Generative Pre-Training) is a classic Autoregressive language model with Transformer as bottom implemented mechanism [49]. GPT uses task-related inputs to improve migration efficiency with a slight change of the model structure to fit different tasks that eliminates the suffering of modifying the architecture for a particular task.

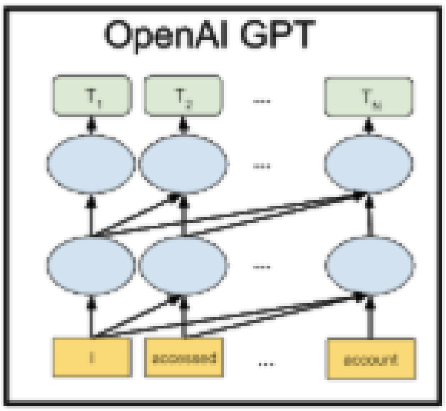


Figure 3.2: Transformer based model architecture of OpenAI GPT [10]

Effectiveness of GPT was verified on totally four language understanding tasks namely natural language inference, question answering, semantic similarity, and text classification [49]. The paper also pointed out that the Transformer (see in Figure 3.2 for model architecture) is noteworthiness and has an eminent interpretation for Natural Language Inference tasks compared to LSTM which used in ELMo.

3.1.3 BERT

BERT[1] (Bidirectional Encoder Representation from Transformers) [10] is another manifestation of transformer's superiority. As demonstrated in Figure 3.3, deep bidirectional Transformer encoder is used in model architecture with more effectiveness than either a one-way model (e.g. GPT) or a two-way model (e.g. ELMo) with the shallow concatenation due to self-collision might happen indirectly in a multi-layered context.

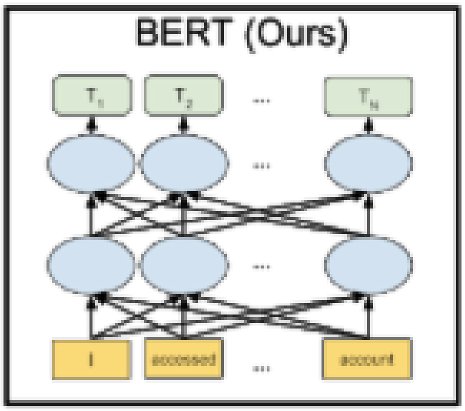


Figure 3.3: Model Architecture of BERT with bidirectional Transformer [10].

Motivated by Cloze task [54], Masked LM (MLM) is an approach that masks a portion of input tokens randomly and uses the transformer encoder to predict it, resulting in a distributional contextual representation under compulsion [10]. For details, during training, 15% of the tokens in the sequence will be masked and predicted, rather than predicting each token as CBOW does. However, a conflict arises during the fine-tuning, that is, we can't encounter mask which artificially introduced in training. The trick method here is to replace 10% of the words into the other tokens in vocabulary, and another 10% of tokens remain unchanged, while the remaining 80% are replaced with [mask]. Random substitution also occurs in the level of sentences. Under the pre-training of a monolingual corpus, the next sentence is replaced by 50% of the current input, so as to obtain the semantic relationship between sentences. This method is called Next Sentence Prediction.

These two methods can be considered as multiple emergencies in the theory of biological evolution, and simulate various factors to increase the training difficulty (mask+random next sentence) as well as training costs and time loss. However,

[1] `goo.gl/language/bert`

after a rigorous ordeal, we obtain a more interpretable model with an outstanding capability to capture information. Broken 11 new state-of-the-art records, BERT has demonstrated its superiority in most every sub-aspects [10] showing Transformer is a relatively efficient model that captures longer distance dependencies.

3.1.4 XLNet

As the two most advanced pre-training models, XLNet is often used to compare with BERT. Different usage patterns performed in pre-training and fine-tuning phase might cause a performance decrease in BERT. To be specific, the additional [Mask] tag is added in the former phase while is absent in later [60]. Besides, XLNet denies an overly simple assumption proposed by BERT that the masked signals, which to be predicted during the pre-training phase, are independent of which other transparent signals (unmasked). XLNet considers this interrelationship and claims to model this high-order, long-term dependencies which are prevalence in text processing [8, 60].

XLNet is a permutation-based Language Model (see in Figure 3.4), embodying the contextual information in the subsequent sequence that each position is able to exploit macroscopic information. Permuting the factorization order, the predicted position i-th are allowed to get a glimpse of varied content of context. For example, 2-rd, 4-th, 1-st position are given to 3-th as input under factorization orders $2 \rightarrow 4 \rightarrow 3 \rightarrow 1$ in the upper left sub-figure of Figure 3.4.

with the help of Transformer and two-stream self-attention, XLNet maximizes the expected log-likelihood of a source sequence after exhaustion of all possible permutations of the factorization order ($T!$ in total for a length T), which is essentially still an Autoregressive model. Two-Stream Self-Attention involves the content stream, which is a typical calculation process in Transformer, and the query stream used to replace the [Mask] tag that was added by BERT, playing the same role as masking with a distinctive implementation mechanism.

Borrowing the idea of AR model, XLNet can capture bidirectional contextual information in the model while maintaining the forwarding way from left to right, reflecting the innovation point of XLNet. However, the independent state between mask tokens can be compensated by training data with an astronomical size which is not lacking in both BERT and XLNet model.

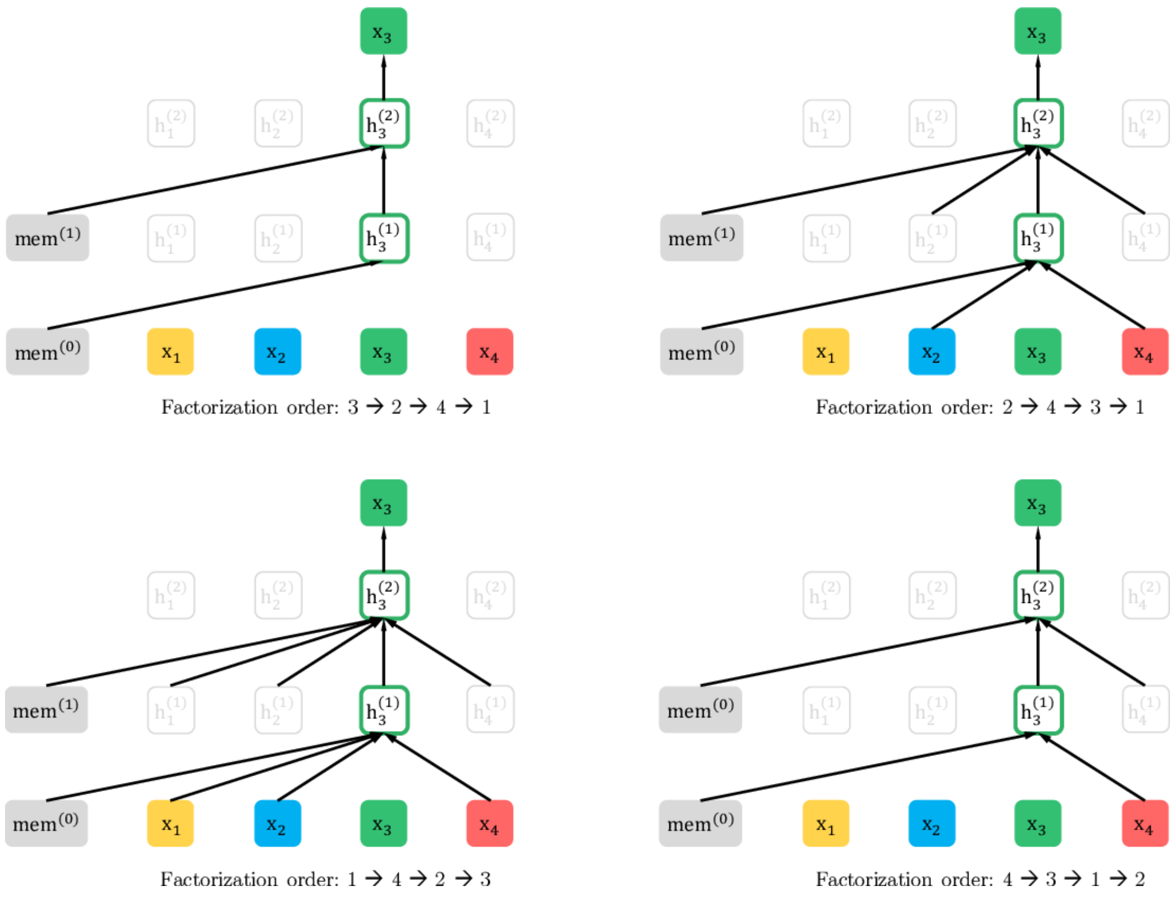

Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2
mem$^{(0)}$
mem$^{(1)}$
x_1
x_2
x_3
x_4
$h_3^{(1)}$
$h_3^{(2)}$

Figure 3.4: Permutations of the factorization order [60]. The i-th tokens of original sequence could peer into the front information based on generated factorization order ignoring original order.

3.2 VADER

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a sentiment analysis tool based on syntactic and semantic rules [23]. It shows significant performance on sentiments expressed on social media such as Twitter because it is specifically attuned for Internet parlance including slang(e.g. “Nah”), acronyms (e.g. “LOL”), Emoji and Emoticon (“:)”).

According to some shortcoming of Machine Learning algorithms that might generate features in a black box that is unintelligible for human, Hutto and Gilbert use a Wisdom-of-the-crowd (WotC) approach [52] to create a new work-bank for social media. They qualitatively collectd the intensity ratings from human raters with a floating number from -4 (negative) to 4 (positive) where median zero represent

Neutral.

After scrutinising all crawled tweet samples from two human experts independently, a heuristics algorithm was proposed by isolating fives linguistic features to performer all probabilities to sentiment intensity. As reported, punctuation, capitalisation, degree modifiers, the contrastive conjunction and negation, these five expandable features show a variety of sentimental relationship. This generalizable sentiment lexicon based on human-curated gold standard performs quite well at the text of social media as well as other three domains (e.g. movie reviews, technical product reviews and opinion news articles) compared to seven other credible opinion mining lexicons. This method discloses the use for reference and its enlightenment to our project, requiring no training data.

We used VADER to label crawled tweet automatically, resulting in a valid database for ternary classification (details in section 5.2.3). Nevertheless, it is a sentiment analysis method based entirely on an emotional lexicon that is difficult to understand the true meaning of semantics. In the face of some out-of-vocabulary (oov) words, it will become unintelligible. For the word “wow”, such an informal word but is commonly used in social media such as Twitter, VADER gives an appropriate judgement as positive. But for a simple variant, ”wooow’ becomes a factor that plagues VADER, being recognized as neutral. This problem caused by the lack of an extensive vocabulary needs to be considered for both Statistics-Based Methods and Lexicon and Rule-Based Methods.

Chapter 4

Proposed Model

In this chapter, our proposed models are introduced in details. Aiming to capture the contextual information of the input sentence and extract it for sentiment analysis. Using BERT pre-trained model as an extractor, several models are constructed with a variety of different classifiers. For plain text and non-text information (emojis and emoctions), we modelled them separately, and generated a vector representation with the same dimension, and proposed the BERT Origin, BERT CNN model based on plain text. Based on the contextual attention mechanism, we train an Emoji-based sentiment classifier by attending on Emoji embeddings.

4.1 Text Classification

Sentiment analysis is a small branch of text categorization that has almost identical modelling goal that accepts an input and falls into diverse categories (determining different emotions, for sentiment analysis), according to the representation information learned during the modelling.

Being able to reasonably infer and make correct emotional judgements based on natural language input is a troublesome task, which requires a large amount of corpus and training costs, and is even more difficult for such complex application scenarios as Twitter. However, through the prior knowledge provided by the pre-trained model, our model can basically understand the world's language application patterns, and it is easier to capture the statistical clues of effective analysis.

Our major approach is using the pre-trained model with parameter tuning for the specific downstream task – sentiment analysis. BERT provides several large pre-training models that can handle multiple languages, which was adopted by the base architecture. Under 12 layers of self-attention based Transformer encoders, the proposed model is more expressive and more context-aware, improving the ability to capture valuable information.

4.1.1 Word Representation

The primary step is to accept and define language signal such as characters and words, without the vision and auditory sensation as a human. A reliable dictionary and word embedding representation method is considered for the purpose of sentiment analysis.

Special effects of capitalization on semantic understanding and emotional expressions are noticeable, such as full-text cased sentence delivery emphasis, dissatisfaction or other strong emotions. So, we used a cased vocabulary containing 28994 tokens, including English words, special characters that commonly used in natural language processing. Some artificially added tags are also involved, such as [MSK] for opacity prediction, [PAD] for sequence filling to max length, and [UNK] for marking unrecognizable tokens. Word segmentation is to convert the input single long sentence into a sequence consisting of the above tokens, which we hand over to the tokenizer. Besides that, we added 55 high frequency Eomji and two emoction (i.e. ":)" and ":("), to the original dictionary, increasing the size of vocabulary to 29051.

Based on the greedy Forward Maximum Matching Segmentation algorithm from WordPiece [58], the extra description of words in prefix and suffix that commonly used English grammar is effectively processed. Taking the word "interpretable" for example, the word is not included in the dictionary but can be interpreted into two consecutive split tokens as "interpret" and "##able" by the tokenizer, extracting root and suffix representing its adjective attributes. In details, firstly check the word whether matches the tokens in the dictionary, then traversing character by character from the tail, finally get the longest word in front to match the existing dictionary. This matching algorithm allows the model to accept derivative words as well as unusual words, expanding the model applicability.

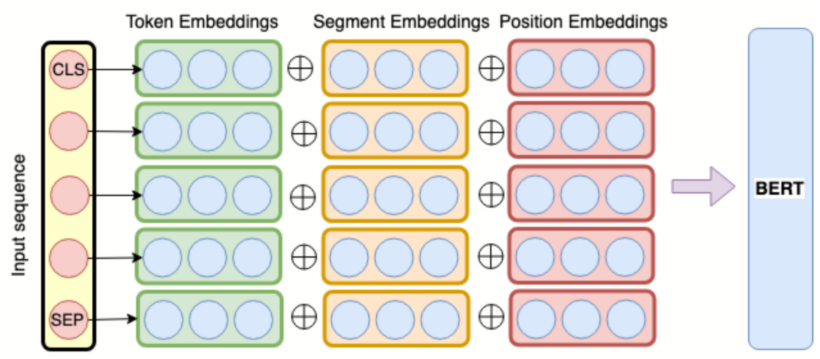


Figure 4.1: Word Embedding for pre-trained model

Figure 4.1 shows the workflow of generating the word embeddings summed by three independent embeddings, namely token, segment and position embeddings. The special tags [CLS] and [SEP] indicate the start and termination of a sentence or separation for document-level input. Tokens embedding is a 768-dimensional vector for each word generated by the matching of the tokenizer. To be specific, we stand on the shoulders of Devlin et al. who unveiled pre-trained token embeddings on the 3300M corpus which was originally generated by random initialization.

Segment embedding acts as a split sentence, which benefits for tasks, such as context matching but mismatches in our sentiment analysis with a single sentence as input. The extraction of the position signal is implemented by positional embedding, compensating the lack of sequential information in the self-attention mechanism. Categorically, the different between ”I love him” and ”he loves me” is evident because of subject-verb agreement. However, “she loved it” and ”it loved she” is confusing, and position embedding avoiding misunderstands caused by positional relationship, while it is superfluous under some time series models such as RNN.

In addition, we unified the input of varying lengths to a fixed length L as 150, in consideration of the upper limit of tweets (limit 140 but double up in 2018), obtaining a vector $\boldsymbol{x} \in \mathbb{R}^{L \times D}$ where $D = 768$ denotes dimension.

4.1.2 Self-Attention Encoder

Similar to cloze in secondary education, BERT uses the mask mechanism to enhance the model that fill in the artificial blanks (or correction) in accordance with contextual information. In this way, the model’s understanding of semantics can be enhanced (detailed in section 3.1.3). While it is unnecessary for our to be fine-tuned model after pre-training on multiple large corpora, including BooksCorpus [63] and English Wikipedia [10].

As shown in Figure 4.2, for the purpose of fine-tuning, we employed the encoder of the BERT 12-layer transformer consisting of a self-attention layer and an intermediate layer (also known as feed-forward). It can be superimposed effectively with the same form of input and output, playing a role as a black box that converts the input word embeddings into an enhanced semantic vector of the same length.

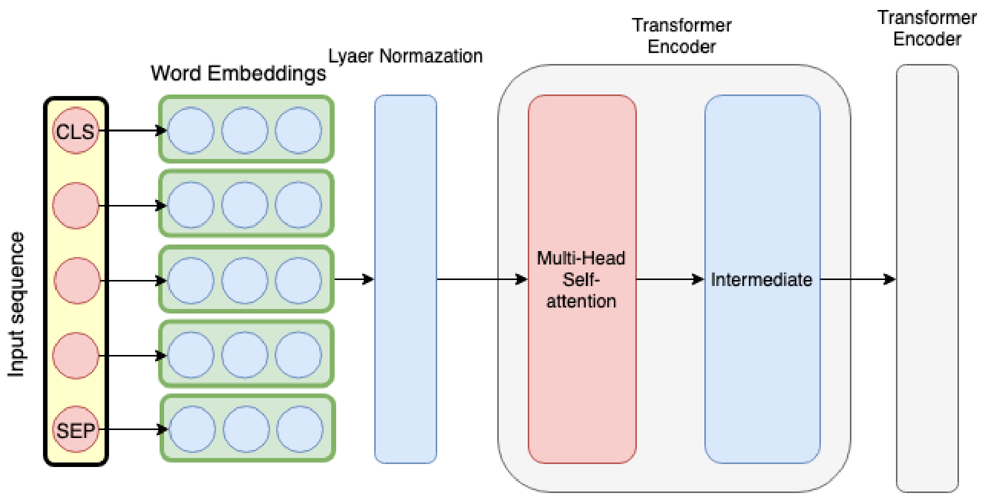


Figure 4.2: Architecture of Transformer encoder.

The self-attention, which as discussed in section 2.6, is a mechanism that calculates the association between each word(query in the illustration) and all other words, and a high multiplicative attention score inevitably represents a high degree of relevance as Equation 2.6.3 [56]. We repeat it as below:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where $d_K = d_Q = d_V = D/m$, m denotes the number of heads and D is the dimension of the word embeddings. The selection of d_k usually depends on the size of number of heads used in multi-head mechanism that attempts to use linearly superimpose on multiple attention scores and obtain global attentions to avoid the long-range disappearance raised by small query size. To specific, weights matrix $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ are learnable parameters that used to map the original Q,K,V (i.e. hidden state of encoder) to lower dimensional spaces as:

$$head_i = \text{Attention}(\boldsymbol{QW}_i^Q, \boldsymbol{KW}_i^K, \boldsymbol{VW}_i^V) \tag{4.1}$$

where $\boldsymbol{W_i}^Q \in \mathbb{R}^{D\times d_Q}, \boldsymbol{W_i}^K \in \mathbb{R}^{D\times d_K}, \boldsymbol{W_i}^V \in \mathbb{R}^{D\times d_V}$ split $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ into different small pieces according to their original position i-th. Total attention score is the concatenation of these m-head attentions, while in practice, we use the wight matrix $\boldsymbol{W}^Q \in \mathbb{R}^{D\times D}, \boldsymbol{W}^K \in \mathbb{R}^{D\times D}, \boldsymbol{W}^V \in \mathbb{R}^{D\times D}$, and then transform the dimension of $\boldsymbol{QW}^Q, \boldsymbol{KW}^K, \boldsymbol{VW}^V \in \mathbb{R}^D$ into $\mathbb{R}^{m\times d_k}$ to calculate attention scores before back to the original size.

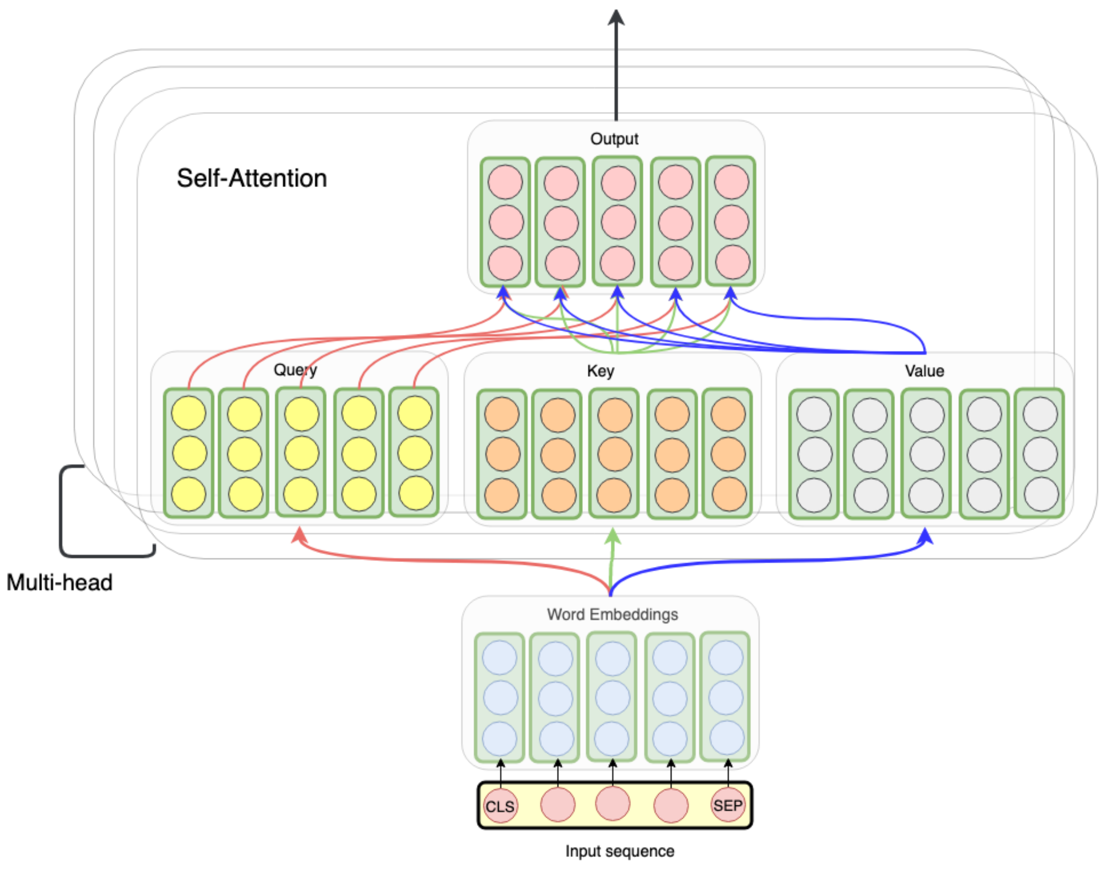


Figure 4.3: Self-attention mechanism.

The intermediate between self-attention layer and next encoder layer is consist of activation function and layer normalization as Figure 4.4 depicts. After a linear expansion, we applied gelu (Gaussian error linear units) [19] as the activation function that adds nonlinear transformation through random regularization, which enhances the generalization ability of the model, conveying the probability description of input.

$$\text{gelu}(x) = xP(X <= x) = x\phi(x) \tag{4.2}$$

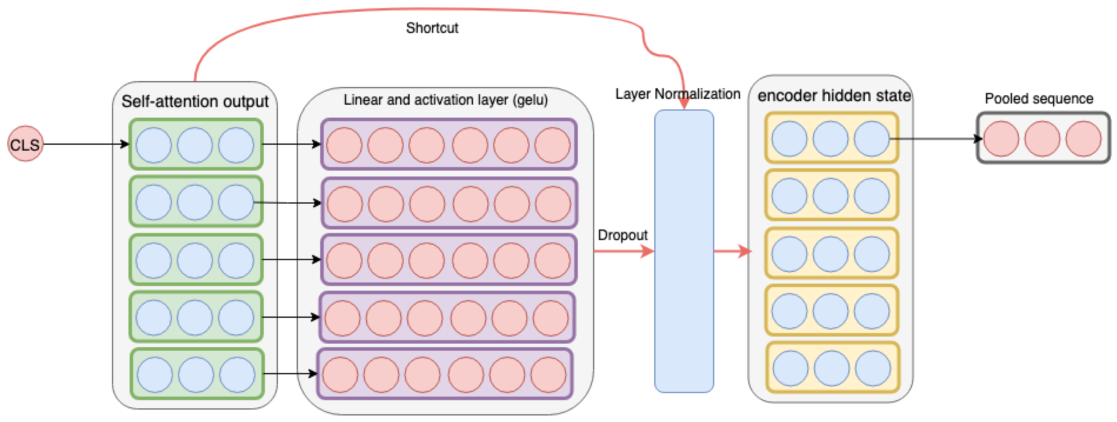


Figure 4.4: Intermediate layer (feed forward)

$$\mathrm{gelu}(x) = 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right) \tag{4.3}$$

where ϕ is standard Gaussian distribution as $N(0,1)$. In implementation, we used its approximate variant as Equation 4.3.

Before sending to the next layer, the non-linear result experienced dropout that discards some entries temporarily with a certain probability, performing ensemble learning that is training different sub-networks on each mini-batch to avoid over-fitting. We summarized the original hidden and the activation results missing some neurons after being randomly selected, then sent the summarization to the layer normalization layer, reflecting the concept of the residual network proposed by He et al.. This simple mechanism, also called shortcut connection, allows the stacking of deep networks and using the identity mapping to survive the long journey without forgetting the starting point.

Also, appearing before the first encoder and following with the activate function, layer normalization is crucial for this series of encoder layers. As mentioned earlier, the encoder has the same form of input and output, which can be seen as a black box to enhance the ability to capture semantic information. Layer Normalization calculates the mean and variance on each layer, ensuring that the output of the encoder has the same statistical distribution, not just dimensions. Thus the hidden states of each layer can be extracted to represent the latest word embeddings. As seen in Equation 4.4.

$$LN\left(x_i\right) = \alpha \times \frac{x_i - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}} + \beta \tag{4.4}$$

where μ_L and σ_L denotes the mean and variance of each layer, respectively. α and β are learnable parameters.

In addition, the first tokens in the input sequence ([CLS]) are used to represent the understanding of the whole sentence. Intuitively, it denotes the special tokens tagging the start containing no specific semantic, unlike other tokens that have actual language understanding. A pooled result (denoted as pooled sequence) with dimension $\mathbb{R}^D$ can be used in subsequent classifiers. One of the proposed model Original BERT (O-BERT) is consist of a stack of 12-layer transforms and a linear classifier that uses the pool sequence as input, while BERT CNN considers all hidden states of the target sentence with multi-convolution layers as classify.

4.1.3 Classifier

Linear Classifier

We simply use linear mapping to convert the pooled sequence of the pre-trained model into a vector with a dimension of the corresponding number of classes (Figure 4.5). Meanwhile, a probability distribution for these two categories can be computed by log-softmax function, while the standard softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined as

$$\sigma(\boldsymbol{z})_i = \frac{e^{\boldsymbol{z}_i}}{\sum_{j=1}^{K} e^{\boldsymbol{z}_j}} \text{ for } i = 1, \ldots, K \text{ and } \boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K) \in \mathbb{R}^K \tag{4.5}$$

However, the log-softmax function is not used separately. In the PyTorch framework that our model built on, log-softmax is automatically computed when invoking cross-entropy, before negative log-likelihood.

The BERT pre-trained model combined with linear classifier is called Original Bert (O-BERT). We can also propose another basic model (named as Frozen Original BERT) that takes BERT as a complex word vector without any fine-tuning, and simply send it to a linear layer for direct classification. The origin of the name comes from the essence of the training process that it freezes all the parameters in the BERT 12 transformer encoder layers, and only fine-tunes the matrix of the linear mapping.

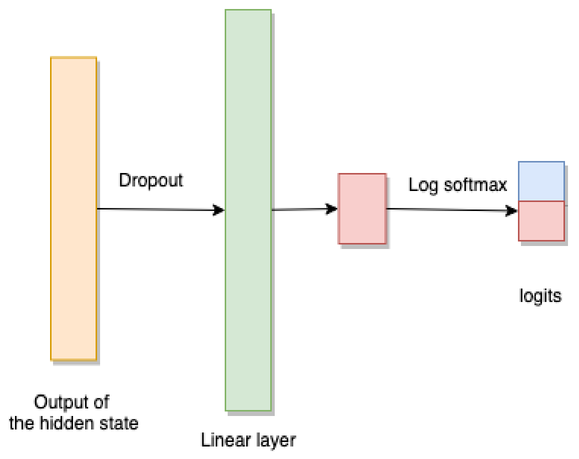


Figure 4.5: Linear Classifier

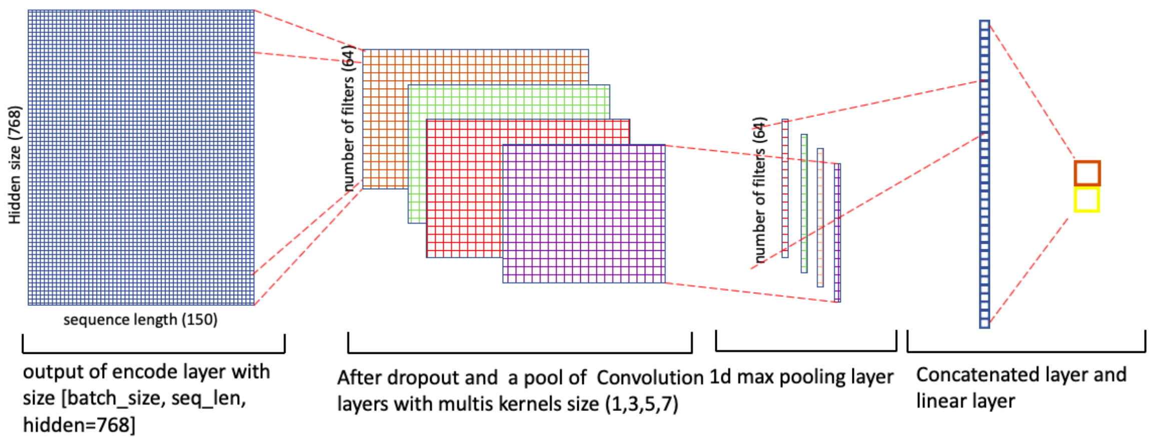

Hidden size (768)
sequence length (150)
number of filters (64)
number of filters (64)
output of encode layer with size [batch_size, seq_len, hidden=768]
After dropout and a pool of Convolution layers with multis kernels size (1,3,5,7)
1d max pooling layer
Concatenated layer and linear layer

Figure 4.6: Use a pool of convolution layers as classify

Convolutional Classifier

Similarly, we extract the output of encoded layers from a pre-trained model and add CNN pool consisting of multiple CNNs with various kernel sizes (1,3,5,7,9 in practice) between the final linear output layers.

The architecture of this CNN-based classifier can be extracted as an independent model to perform the task of sentence classification [25], often referred to as TextCNN. Basic TextCNN model accepts the word embedding sequence as input, denoted as $\boldsymbol{M} \in \mathbb{R}^{L\times k}$ for a k-dimensional word dimensional representation with sequence length L. In general, some public word representation like Word2vec [33] and GloVe [45] are serviceable, while we used the output of the final encoder from a pre-trained model with the same dimensionality.

As seen in Figure 4.6, after several 1-D convolution layers for each word of sequence, we obtained a feature matrix for each layer with size $n \times L - h + 1$ where n, h denote the number of filters and the size of convolution kernel (moving window). This process produces multiple feature matrix with various size depends on the size of kernels. To harmonize these unequal features matrix, we utilise a 1-D max-pooling layer using a stride step with a size of $L - h + 1$, obtaining an in-variate vector concatenated by the maximum values for each feature matrix. We applied dropout after the received vector from the encoder and the concatenated vector that allows our classify to overcome overfitting by ensemble learning. Finally, we obtain a binary vector after a linear mapping, which is identical as basic linear classify.

This TextCNN based classify also used to sentiment analysis independently by frozen the parameters of the pre-trained model and only use the word embedding, which called a Frozen CNN-based BERT (FCNN-BERT) model. As for the corresponding model that needs to be fine-tuned for the entire parameters, we call it as CNN-BERT.

4.2 Emoji Attended Sentiment Analysis

Emoji plays a very important role in online social networking such as twitter, providing us with more interactive information by imitating facial expressions. However, it is biased to determine the sentiment just by Emoji due to the interaction with the literary words, showing some unique semantics, even though have different representations from the text. Taking it as emotional annotation or an emotional feature have been the two most successful treatment [9, 12, 62].

Extracting Emoji and analysing its impact on sentiment analysis is the original intention of designing this attention-based model. Followed by a Transformer encoder that well trained by text, we apply attention mechanism between the hidden states of last encoder layer and Emoji embeddings inspired by Chen et al. who uses LSTM as the encoder.

As seen in Figure 4.7, we extract Emoji from the original tweet, ignoring its original position and keep the context of the text. After the 12-layer Transformer encoder, we obtain a series of hidden states denoted as $\boldsymbol{h}_t$ for each token in original sequence by fine-tuning with the training dataset.

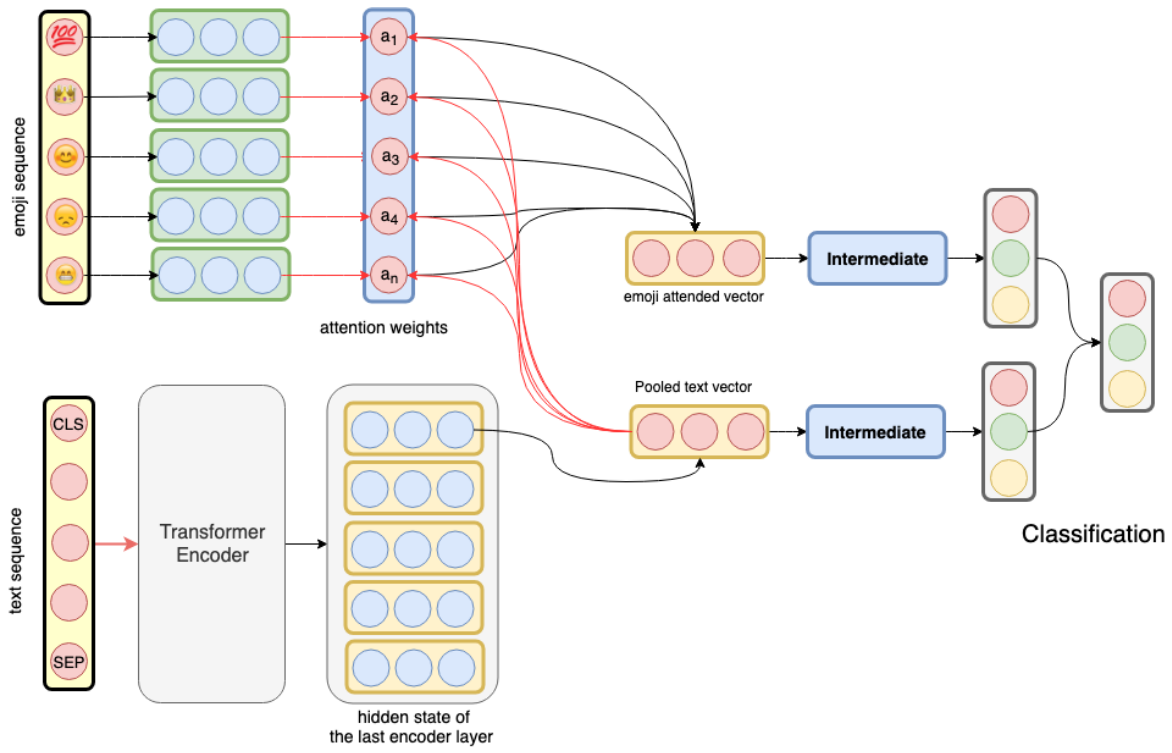


Figure 4.7: Architecture of Emoji attended model

The initial approach was to calculate the attention weights between Emoji embedding and each hidden state, considering the interaction relationship brought by Emoji based on text semantics as

$$\alpha_t = \frac{\exp\left(Attn_{t,i}\left(\mathbf{w}_t, \mathbf{e}_i\right)\right)}{\sum_{i=1}^{m} \exp\left(Attn_{t,i}\left(\mathbf{w}_t, \mathbf{e}_i\right)\right)} \tag{4.6}$$

Where $\boldsymbol{h}_t \in \mathbb{R}^D$ and $\boldsymbol{e}_i \in \mathbb{R}^D$ represent the t-th hidden state of text and the i-th Emoji embedding. $Attn_{t,i}$ are used to calculate the attention score between $\boldsymbol{h}_t$ and $\boldsymbol{e}_i$ by dot production as

$$Attn_{t,i} = \boldsymbol{h}_t \boldsymbol{e}_i^T \tag{4.7}$$

Concatenated with the original hidden states, we obtain a new input $\boldsymbol{v_t} \in \mathbb{R}^{2D}$ for classification as

$$\boldsymbol{v}_t = [\boldsymbol{h}_t, \alpha_t \cdot \boldsymbol{h}_t] \tag{4.8}$$

In the approach proposed by Chen et al., the hidden state is replaced by the original word embedding and sent this splicing vector as input to LSTM. However, we have used the transformer encoder to achieve the initial training of text embedding, ensuring the hidden state can be regarded as the new word embeddings that matching the distribution of the current dataset.

Considering the first token [CLS] as the summary, we reconstruct the model called Emoji Attended with BERT (EA-BERT) that no longer use all the hidden states globally, reducing the amount of computation, as shown in Equation 4.9

$$\alpha' = \frac{\exp\left(Attn_i\left(\mathbf{u}, \mathbf{e}_i\right)\right)}{\sum_{i=1}^{m} \exp\left(Attn_i\left(\mathbf{u}, \mathbf{e}_i\right)\right)} \tag{4.9}$$

where $\boldsymbol{u}\ inR^D$ denotes the first token. Therefore, we get the new input to the classifier, which is the same as 4.8:

$$\boldsymbol{u}' = [\boldsymbol{u}, \alpha' \cdot \boldsymbol{u}] \tag{4.10}$$

To verify the different expressiveness of text and Emoji, we create two independent classifies and use relu as the activation function in the Intermediate layer as seen in Figure 5.9. The combination weights can be learnable parameters or a fixed matrix by an artificial decision.

Chapter 5

Experiments

In this chapter, we described the details of the basic pre-trained plain text model and the Emoji-based attention model. we run our model on different datasets, comparing the performance of various training methods, including loss function, optimizer, schedule, and their hyper-parameters. We verified the ability of the pre-trained models to capture semantic information and fine-tuned them into various classifies to fit sentiment analysis.

5.1 Evaluation Methods

The most basic measurement – accuracy, is the proportion that labelled data correctly classified as the of the corresponding category. However, for some particular datasets, the applicability of the model cannot be accurately evaluated by accuracy as the unique metric. One extreme example is an imbecile model for cancer diagnosis with the only output of “health”, resulting in high accuracy in practical applications owing to the majority do not carry cancer cells. It is a confusing model with inadequate classification capacity that does not match our expectations and needs to be optimized, inspiring the introduction of F1 score.

$$
\begin{aligned}
\text{accuracy} &= \frac{TP + TN}{TP + FN + TN + FP} \\
\text{precision } &= \frac{TP}{TP + FP} \\
\text{recall } &= \frac{TP}{TP + FN} \\
\text{F1 score} &= \frac{2 \times \text{ precision } \times \text{recall}}{\text{precision} + \text{recall}}
\end{aligned}
\tag{5.1}
$$

The above equation 5.1 shows the difference of the computing method between accuracy and F1 score as well as precision and recall of which TP, TN, FP, FN represent the True Positive, True Negative, False Positive and False Negative respectively [29]. The precision can be interpreted as whether the amount of related information retrieved is accurate, while the model coverage that whether the retrieved data is comprehensive is given to the recall to judge. F1 score is an aggregative indicator that considers both precision and recall and returns their harmonic mean. An effective equilibrium of exactitude and coverage is what F1 score wants to measure.

Above discussion is based on binary classification, Two expanded F1 score evaluation – macro F1 and micro F1 are available when the number of categories increases. The calculation method of the macro-average F1 is to separately calculate the F1 value for each type (e.g. positive and non-positive) and then take the arithmetic mean of these F1 values as a global metric. For micro-average F1, Like macro-average F1, first split an n-class problem into n binary problems. Then use the average precision and recall for these n binary problems to calculate the global micro-average F1. Macro F1 is used in following experiments because it is more effective considering the distribution of data categories.

5.2 Dataset

5.2.1 Public Dataset

Sentiment140 is a resource dataset published by Stanford [13]. Containing 1.6 million English Tweet categorised into two labels – positive and negative, not as Kaggle claimed that exist a third label as neutral in their competition website [1]. An Au-

[1]https://www.kaggle.com/kazanova/sentiment140

Emojis	Occurrences	Sentiment			
		Positive	neutral	negative	compound
😂	14622	0.247	0.285	0.468	0.221
❤️	8050	0.044	0.166	0.790	0.746
♥️	7144	0.035	0.272	0.693	0.657
😍	6359	0.052	0.219	0.729	0.678
😭	5526	0.436	0.220	0.343	-0.093
😘	3648	0.053	0.193	0.754	0.701
😊	3186	0.060	0.237	0.704	0.644
👌	2925	0.094	0.249	0.657	0.563
💕	2925	0.042	0.285	0.674	0.632
👏	2925	0.104	0.271	0.624	0.520

Table 5.1: Top 10 Emojis in 13 European Language

tomatic annotation method was used in their report that simply divides sentiment by emoticons, like ":)", ":(". It still supports our model training, although in an uncomplicated way of data segmentation.

As a vital part of the emotional expression of social media such as tweets, Emoji is a series of graphic characters including facial expressions, weather, transportation and other daily life elements, numbers and national flags, etc. In 2015, under the help of 83 human annotators, an Emoji lexicon was released [38], learning the statistics of over 70,000 tweets in 13 European languages, in order to analyse the Emojis occurrence and its emotional patterns. The table 5.1 above shows the ten Emojis with the highest occurrences as well as its sentiment and Unicode essences, the completed form can see at Emoji Sentiment Ranking v1.0 [2].

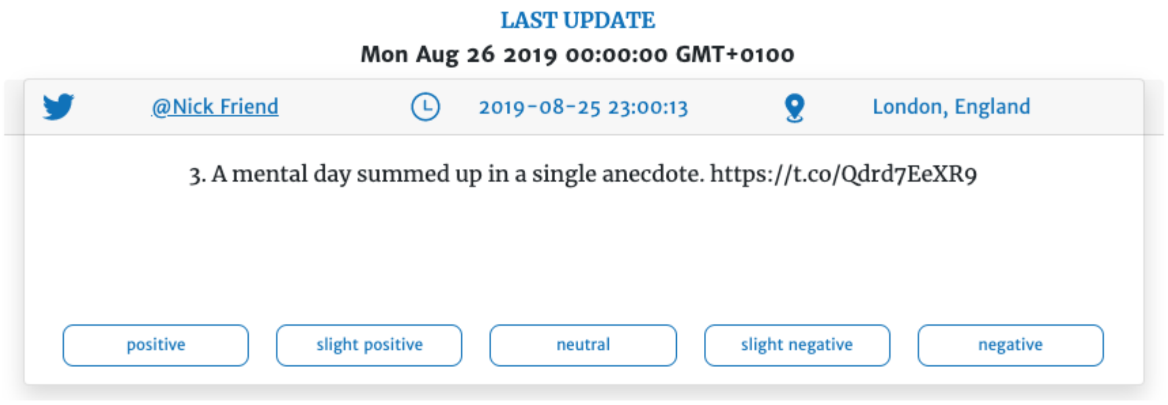
5.2.2 Manual Annotation

Admittedly, Tweet is often mixed with URL links, images and other multimedia to enrich the diversity of emotional expression. Emoji, as the most commonly used form of twitter, conveys multiple meanings besides graphical symbols. However, in previous studies, complex elements such as Emoji were often removed. To better

[2]http://kt.ijs.si/data/Emoji_sentiment_ranking/

understand the actual use of tweet, a webapplication[3] is settled up to collect and annotate tweets manually by our kind-hearted visitors.

As seen in Figure 5.1, five available buttons can be selected, depending on visitors' great true feelings about the tweets shown above. "Slight positive" and "slight negative" as two additional tones compared to traditional three-classification problem, allows the annotators to give appropriate judgement when encountering some tweets that are emotionally vague and not easily determined.


LAST UPDATE
Mon Aug 26 2019 00:00:00 GMT+0100

@Nick Friend 2019-08-25 23:00:13 London, England

3. A mental day summed up in a single anecdote. https://t.co/Qdrd7EeXR9

Figure 5.1: Web Application Interface for Data annotation

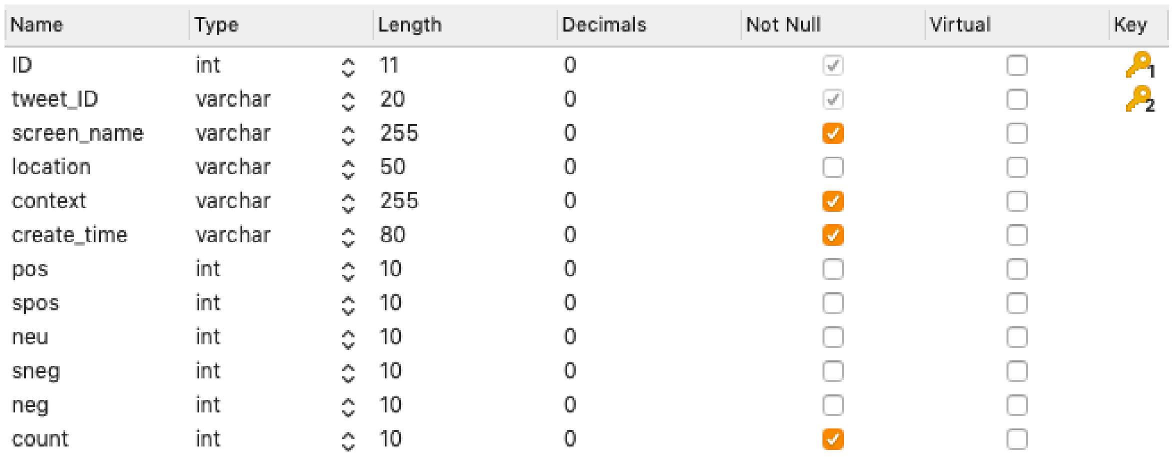
20 tweets are updated daily at midnight local time in London (British Summer Time as known as GMT +0100), 16 of which are collected from the last day of the update time. Divide 24 hours a day into 8 segments of three hours, assigning two quotas of tweets to each segment, that is $8 \times 2 = 16$ in total. The remaining four are involved in the data collection as validation, selected from sentiment140 dataset. Then, we accumulate the difference between the accepted value and the ground truth of the validation, and effectively filter the whole twenty tweets of same annotators under the threshold processing. Besides that, we computed the time interval for the whole process of rating, the total time-consuming also become one of our measuring standards to decide keep or eliminate tweet from some demotivating online volunteers. The proposed collection method is useful for acquiring a reliable dataset, increasing the confidence degree of data as well as user engagement.

In the above step, we completed the basic screening for crawled twitter as following: 1. filtered the retweets starting with 'RT', which normally spread the message or express support and approval for its forwarded content. 2. removed replies because

[3]`http://testbeforetweet.uksouth.cloudapp.azure.com`

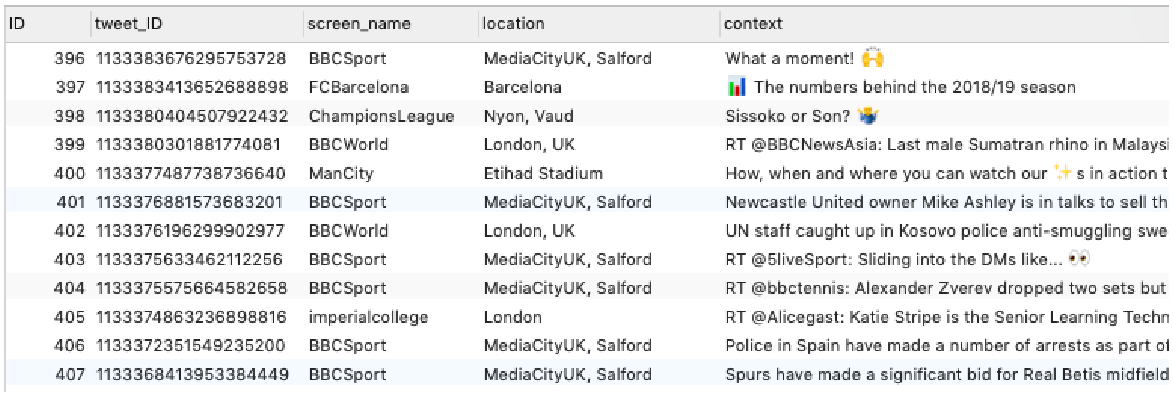
it is usually a response to another tweet. It is difficult to judge the sentiment of reply without knowing the content of the original subject.

Shown in Figure 5.2, fetched data stores in a MySQL database which cotains a structured table with field of ID, tweet_ID, screen_name, location, context, created_time, pos, spos, neu, sneg, neg, count,while the last sixth fields represent five return results of rating and the count number of total rating. Navicat GUI[4], the visualisation tool we used in this project, providing a manifold way to control this MySQL database.



Name	Type	Length	Decimals	Not Null	Virtual	Key
ID	int	11	0	✓	☐	🔑1
tweet_ID	varchar	20	0	✓	☐	🔑2
screen_name	varchar	255	0	✓	☐	
location	varchar	50	0	☐	☐	
context	varchar	255	0	✓	☐	
create_time	varchar	80	0	✓	☐	
pos	int	10	0	☐	☐	
spos	int	10	0	☐	☐	
neu	int	10	0	☐	☐	
sneg	int	10	0	☐	☐	
neg	int	10	0	☐	☐	
count	int	10	0	✓	☐	

(a) Structure of table status



ID	tweet_ID	screen_name	location	context
396	1133383676295753728	BBCSport	MediaCityUK, Salford	What a moment! 🙌
397	1133383413652688898	FCBarcelona	Barcelona	📊 The numbers behind the 2018/19 season
398	1133380404507922432	ChampionsLeague	Nyon, Vaud	Sissoko or Son? 🤷
399	1133380301881774081	BBCWorld	London, UK	RT @BBCNewsAsia: Last male Sumatran rhino in Malaysi
400	1133377487738736640	ManCity	Etihad Stadium	How, when and where you can watch our ✨ s in action t
401	1133376881573683201	BBCSport	MediaCityUK, Salford	Newcastle United owner Mike Ashley is in talks to sell th
402	1133376196299902977	BBCWorld	London, UK	UN staff caught up in Kosovo police anti-smuggling swe
403	1133375633462112256	BBCSport	MediaCityUK, Salford	RT @5liveSport: Sliding into the DMs like... 👀
404	1133375575664582658	BBCSport	MediaCityUK, Salford	RT @bbctennis: Alexander Zverev dropped two sets but
405	1133374863236898816	imperialcollege	London	RT @Alicegast: Katie Stripe is the Senior Learning Techn
406	1133372351549235200	BBCSport	MediaCityUK, Salford	Police in Spain have made a number of arrests as part of
407	1133368413953384449	BBCSport	MediaCityUK, Salford	Spurs have made a significant bid for Real Betis midfield

(b) Data samples of table status

Figure 5.2: Visualisation of stored data with its table structure

Starting from 1st of July, the manual annotation ended up on the last day of August. After two months, we only get a small dataset with 603 tweets and 1322 ratings. In order to match the three-category classification in the traditional sentiment analysis,

[4]https://www.navicat.com/

we make some slight modification to options of “slight”. Assign them evenly to two adjacent tones, that is giving 50% statistical data of the “slight positive” to the positive and neutral separately. Finally, the maximum value of each category is determined as the sentiment of the tweet. Removing some isolated tweets with only one annotation, which is too thin with no convincing, we finally obtain 337 annotated tweets containing 85 positive, 98 negative and 154 neutral items. We call this dataset as manual337 for notation.

According to the feedback from our annotators, deciding the emotion of tweets is hard because some of which miss attached multimedia information. In general, an external link and or picture might contain much more information than the text itself. As for the sentences with uncertainty, Human annotator probably chooses neutral to stand in the middle, not either side of the polarity.

5.2.3 Automatic Annotation

Manual annotation shows the disadvantageous of time-consuming and laborious. In addition, we have also adopted an automatic labelling strategy to increase the size of the dataset effectively.

Using Twitter API [5] and a public python library tweepy [6], filtering replies, re-tweets, image-oriented and tweets longer than 140 characters, we grabbed a total of 190,959 valid messages. In order to facilitate horizontal comparison with previous research, we still choose 140 characters as the upper limit of tweet content, even if it has doubled to 280 since November 7, 2017. The crawled tweets are stored with the collation of utf8mb4 which provides a convenient for storing Emoji character. It should be noted that the crawling and storing steps here also applied to the collection of manually labelled data.

VADER-Sentiment-Analysis[7] (mentioned in section 3.2) takes effect as an automatic annotator, sending an output named compound score between -1 (strong negative) to 1 (strong positive) when given a tweet. This score denotes the ranking of sentences computed by the accumulation of valence scores for each word. For example, for twitter expression “What happened today is sux! But I’ll be fine 😊”, a compound

[5]`https://developer.twitter.com/en/docs.html`

[6]`https://www.tweepy.org/`

[7]`https://github.com/cjhutto/vaderSentiment`

score of 0.8671 is given combined with negative (0.079), neutral (0.45) and positive (0.472). The example above shows that VADER can accurately identify some online slangs such as sux, and speculate on its sentiment according to its indecent meaning.

Drawing on the automatic labelling method from Chen et al. [4], we filter the results of this valid sentiment lexicon to enrich the database. For polarized tweets (positive and negative) with an absolute value of the emotional score over 0.6 (0.7 in the original paper), we concur with the judgements given by VADER and regard them as ground truth, since high sentiment scores have a certain degree of credibility to avoid some inexplicit, controversial answers. Similarly, we have chosen a smaller range (-0.05 to 0.05) of scores between which neutral results will be not denied. The above data automatic annotation was evaluated in section 5.5.2

Eventually, after data prepossessing (details in section 5.3.1), we got an adjusted dataset with rich resources of Emoji containing 28,702 positive tweets, 13,176 negative tweets and 68,781 neutral tweets (110,659 in total). We call it ternary Emoji (3-emoji). Considering that many datasets and proposed methods are based on positive and negative two classifications, we remove the neutrality of the dataset and got a different one with only two categories, called as binary Emoji (2-emoji).

5.3 Approach

5.3.1 Data Prepossessing

Character Handling

With an intensive observation of the data we collected, we found that examples carrying with distinct positive emotions such as “Incredible!beautiful techie” are automatically classified as completed neutral with compound score 0, which forces us to delve into the feasibility of VADER. Through many experiments, we found that the space between words has a significant impact on the results. we got an exact and rational result of there above example (see in Table 5.2), after a slight modification to the original data as:

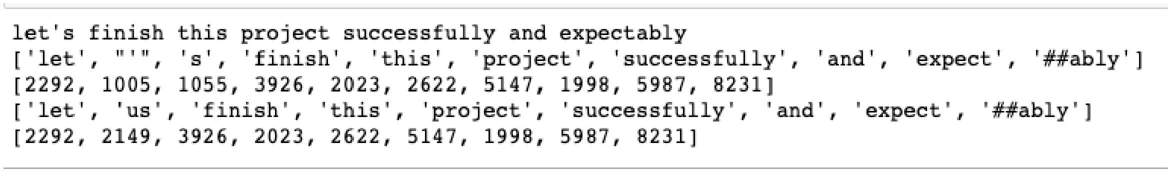
- adding a space after every word, punctuation mark and emotional symbol

content	sentiment score			
	pos	neu	neg	compound
Incredible!beautiful techie	0.0	1.0	0.0	0.0
Incredible! beautiful techie	0.677	0.323	0.0	0.636

Table 5.2: Vader Results Influenced By Space

- remove all special character like “@” for mention, “#” for hash tag of a event
- remove all digits and non-ASCII characters including unescaped line break and carriage-return character.
- remove all URL information containing “www”, “http” or “https”
- remove repeat characters involved in emoticons (e.g. convert “:)))” and “:——(” into “:)” and “:-(”, respectively)

Contraction Mapping



```
let's finish this project successfully and expectably
['let', "'", 's', 'finish', 'this', 'project', 'successfully', 'and', 'expect', '##ably']
[2292, 1005, 1055, 3926, 2023, 2622, 5147, 1998, 5987, 8231]
['let', 'us', 'finish', 'this', 'project', 'successfully', 'and', 'expect', '##ably']
[2292, 2149, 3926, 2023, 2622, 5147, 1998, 5987, 8231]
```

Figure 5.3: Illustration of word segmentation for folded sentence. We expanded the phase “let’s”” into “let us”. Unlike the increase in intuitive understanding, this operation reduces the sentence effective length, restoring semantics of the single letter “s” itself.

As mentioned earlier in section 4.1, using the pre-trained model BERT and the corresponding tokenizer, we can convert the input sentence into a sequence of the corresponding vocabulary id. For some commonly used abbreviations and contractions, we do the work of expanding and restoring (e.g. “I’d”, “ain’t”, “it’s”, etc.). As seen in Figure 5.3, contraction mapping reduces the length of the sentence while eliminating the extra meaning of special symbols. In short, focusing on the text itself allows us to get better training results.

Happy emoticons	Sad emoticons
‘:-)’, ‘:)’, ‘;)’, ‘:o)’, ‘:]’, ‘:3’, ‘:c)’,‘:>’	‘:L’,“:-/’, ‘>:/’, ‘:S’, ‘>:[’, ‘:@’
‘:>’, ‘=]’, ‘8)’, ‘=)’, ‘:}’, ‘:^)’, ‘:-D’	‘:[’, ‘:-——’, ‘=L’, ‘:<’,‘:-[’, ‘:-<’
‘:D’, ‘8-D’, ‘8D’, ‘x-D’, ‘xD’, ‘x-D’	‘=\\’, ‘=/’, ‘>:(’, ‘:(’, ‘>.<’, ‘:’-(”
‘XD’, ‘=-D’, ‘=D’,‘=-3’, ‘=3’, ‘:-))’	‘:\\’, ‘:-c’,‘:c’, ‘:{’, ‘>:\\’, ‘;(’
“:’-)”, “:’)”, ‘:*’, ‘:^*’, ‘>:P’,‘:-P’,‘:P’	“:’(”, ‘:-(’
‘X-P’, ‘x-p’, ‘xp’, ‘XP’, ‘:-p’,‘:p’, ‘=p’	
‘:-b’, ‘:b’, ‘>:)’, ‘>;)’, ‘>:-)’,‘<3’	

Table 5.3: Mapping list for 72 emoticons

Emoji and Emoticon Extraction

Unlike Chen et al., who only constructed a dataset for the high-frequency 55 Emojis [4], we pick all Emojis to the database without filtering. We intuitively believe that uncommon Emojis convey more semantic information than sentiment. An example of this is that 🏠 denotes a house, acting as a content messenger with equivalent expressive power as words. According to this intention, we translated low-frequency Emojis into text by a python library called emoji [8], keeping 55 frequently used Emojis. In addition, we erase the positional information of remained Emojis based on an assumption that it only contains emotional information which is often independent of position.

With regard to emoticon, Go et al. pointed out that could use emotion mapping method to reduce the number of emotions to consider [13]. We expanded the lexicon to 46, 26 emotions for happy and sad emotional character sequence, and unified these multiple emoticons to the most basic ones namely ”:)” and ”:(” See Table 5.3 for the full mapping list of emoticons.

The processed Emojis and emoticons are added to the BERT’s vocabulary, and the corresponding word embedding can be trained in subsequent training, avoiding being identified as unknown or character segmentation. For example “:)” could be tokenized into “:” and “)”, losing its semantic information.

[8] https://github.com/kyokomi/emoji

5.3.2 Focal Loss

Focal loss is a loss function proposed by Lin et al. [27], initially used in the field of object detection. Focal loss is used as a loss function, replacing cross-entropy to enhance training efficiency for the remission of data imbalance appearing on the physical truth of twitter as well as datasets we collect.

The following formula is compared with the focal loss and cross-entropy, which is an acclaimed loss function for classification problems

$$\begin{aligned} \mathrm{FL}\left(p_{\mathrm{t}}\right) &= -\alpha_{\mathrm{t}}\left(1-p_{\mathrm{t}}\right)^{\gamma} \log \left(p_{\mathrm{t}}\right) \\ \mathrm{CE}\left(p_{\mathrm{t}}\right) &= -\log \left(p_{\mathrm{t}}\right) \end{aligned} \tag{5.2}$$

where notation p_t is estimated probability (for binary classification) as:

$$p_{\mathrm{t}} = \begin{cases} p & \text{if } y=1 \\ 1-p & \text{otherwise} \end{cases} \tag{5.3}$$

Two parameters are added into focal loss of which a weighting factor α_t can also appear independently in cross entropy (α-balanced CE) serving as a remedy for data imbalance by weighting different categories. In the experiment we select it as the inverse data density instead of hyper-parameter searching. The focusing parameter γ weaken the impact of correctly classified samples on training by giving an exponential amplification on high p_t, letting model to focus on misclassified data. Here need explanation is focal loss is equivalent to cross entropy when γ equals zero.

Including undersampling and oversampling, sampling is another way to handle data imbalance. By varying the sampling probability, the sample data distribution is changed, inducing permanent distortion of original data. In a word, sampling gives the sample a different possibility and decide whether it appears in the training data. That is, the category with a small number of samples will be more likely to appear, and finally get a balanced training set with the same size as original, regardless of the previous category ratio.

We applied focal loss and weighted sampling with cross-entropy on both balanced data (Figure 5.6) and unbalanced data constructed artificially (Figure 5.7 and Table 5.6) in the section 5.5.3.

5.4 Training

5.4.1 Loss function

During the experiment, we found some disadvantages of weighted sampling (Figure 5.7). moreover, the focal loss (mentioned in the section 5.3.2) also shows disappointed results for our classification problem that is not expected. In this case, we consider α-CE (balanced cross entropy) as loss function, which is equivalent to setting the focusing parameter in focal loss to 0 (Equation 5.2).

5.4.2 Optimiser

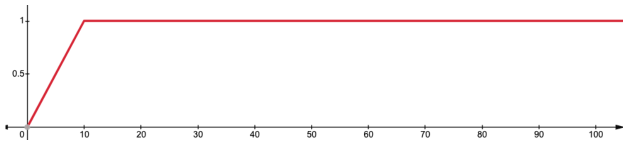
Adam

In general, Adam [26] are considered as an ideal optimiser for most deep learning tasks, interacting with each L2 regularization for over-fitting problem. Nevertheless, Loshchilov and Hutter claimed that weight decay, the regularization implementation of most current Deep Learning frameworks, is coupled to Adam. We uses AdamW [30] as major optimiser, and apply weight decay only on module parameters with exception of the layer normalization and bias. In the fine-tuning phase, we also adopted Stochastic gradient descent (SGD) with momentum to accelerate the convergence of the model.

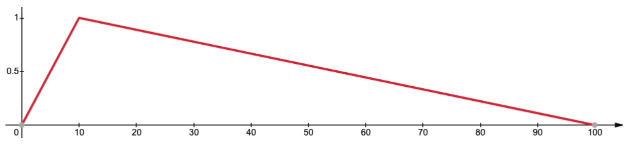
5.4.3 Warm up

Warm up is proven to be an effective training method when faced with complex deep networks [15]. The specific measure is to set a schedule for the gradient descent process, selecting the learning rate according to different stages.

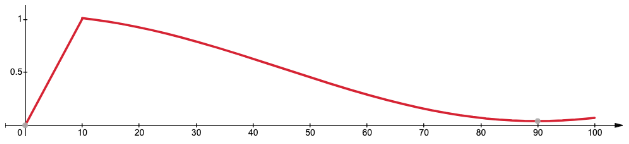
As seen in Figure 5.4, the segmentation is based on the steps for each epoch (x axis) while y axis denotes the scale ratio for main learning rate. Setting a schedule for a given proportion of data, warming up mechanism can scale the learning rate in constant, linear or cosine. Generally, in the early stage of training, a small learning rate is suitable for data distribution correction. While in the subsequent training



(a) Constant warm up schedule



(b) Linear warm up schedule



(c) Cosine warm up schedule

Figure 5.4: Warm up schedules

process, the model has some exact priors for the current mini-batch. In this case, the model deviation might not easily be triggered by a large learning rate so that it can be turned up appropriately. Meanwhile, the warm up can alleviate the violent fluctuation of the model training caused by the large variance of the data distribution in the mini-batch. The smaller learning rate can make the weight more stable which explains the efficiency of the warm up. In our experiments, constant and linear is used, but the appropriate hyper-parameter (proportion) has not been found to fit our model.

5.5 Experiments

5.5.1 Experimental Environment

We settle up a virtual server on Microsoft Azure[9] and used multi-GPUs for parallelization to speeds up convergence. All proposed model and related experiments are implemented on PyTorch [44] which is a mature framework that is suitable for researchers and small-scale projects. The BERT trained model [10] are provided by

[9]`https://azure.microsoft.com/`

pytorch-transformers [10], and we use a PyTorch implementation of focal loss function [11] during the experiments.

Training deep networks on multiple Graphics processing units (GPUs) can speed up convergence, but load balancing is a critical issue that cannot be ignored. Ideally, the training speed should be positively correlated with the number of GPUs, because for each mini-batch of a large dataset, the parallel arithmetic subdivides the mini-batch evenly for every single GPU. However, the basic parallelization still gives the data storage and loss calculation tasks to the main GPU (first as default). This mechanism causes the main GPU to pause because of insufficient memory, leading to the entire training process to abort, while other GPUs are far from reaching the upper limit of the load.

We migrated the loss function calculation to the inside of the model that calculates the loss on each GPU independently. The main GPU is only responsible for distributing and accepting data including input, loss result and even hidden state for external usage. This design slightly alleviates the training suspension caused by load imbalance, allowing us to increase the size of the mini-batch.

For most of the subsequent experiments are tested under the following parameters as:

- Learning rate: 1e-6
- Batch size: 96
- Max length of text: 150
- Max length of Emoji: 16
- dropout: 0.1

5.5.2 Vader Effectiveness

Using manually labelled data manual337 (collection method in section 5.2.2), we check the effectiveness of the our automatic annotation method in section 5.2.3.

[10]`https://github.com/huggingface/pytorch-transformers`

[11]`https://github.com/clcarwin/focal_loss_pytorch`

194 of all 337 records can be classified into three categories, that is, the results returned by Vader are within the range of our delineation. In a word, there same data samples that cannot be divided into either category, resulting in be filtered by Vader.

Considering the results of VADER as predictions and manual337 as ground truth, we evaluated the VADER model with accuracy and F1 score. As shown in Table 5.4. It can be seen that VADER is showing good accuracy, but we are temporarily unable to measure the superiority of this value.

	Number of tweets	Accuracy	F1 Score
binary	75	0.8400	0.829
ternary	194	0.6134	0.5879

Table 5.4: Evaluation of VADER sentiment analysis

The ternary in the Table 5.4 denotes the 3-classes data including three tones (positive, negative and neutral), while binary is 2-classes dataset that remove neutral samples. The results shows that VADER gives a more biased prediction for tweets without polarity, which is also a bit of trouble for human annotators according to their feedback.

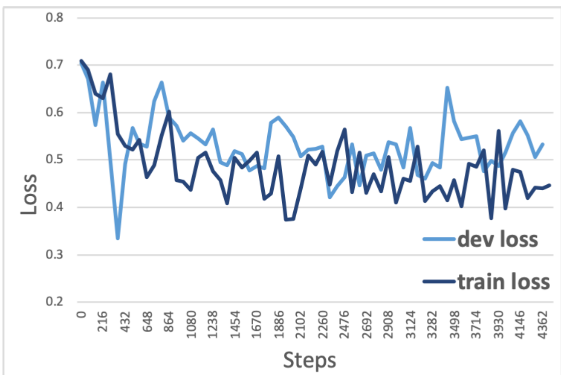
5.5.3 Plain Text Classification

Sentiment140 Dataset

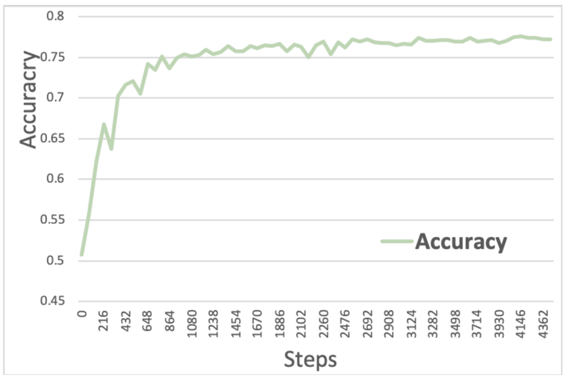
For this binary classification dataset Sentiment140 [13], we mainly completed the performance comparison for focal loss and weighted sampling with on balanced and unbalanced data, and evaluated the CNN-Bert text classification model used in this thesis. However, it is a massive database containing 1.6 million tweets with automatic annotation, which makes it difficult to traverse the entire database completely. In the following experiments, we sometimes only select partial data for model training, reducing the time cost.

According to a corresponding leader board from Kaggle competition [12], We took the 34th position in this ranking with the correct rate of **78.2%** with leader being

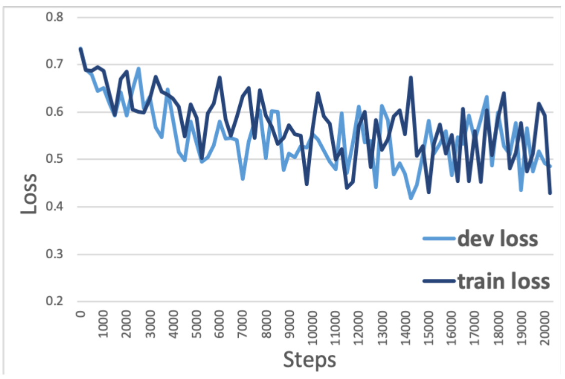
[12] `https://www.kaggle.com/c/tweet-sentiment/leaderboard`



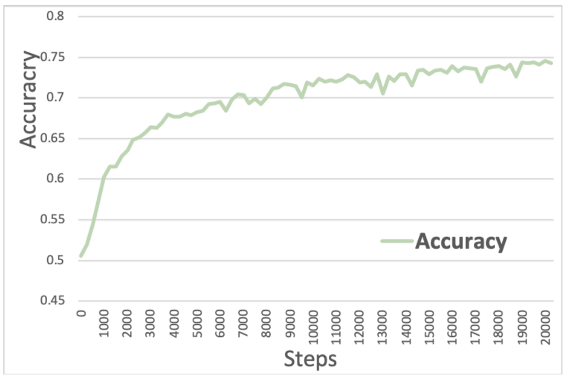
(a) loss of complete traversal



(b) accuracy of complete traversal



(c) loss of local traversal



(d) accuracy of local traversal

Figure 5.5: No advantages for a large dataset. Subfigure 5.5(a) and 5.5(c) shows cognate descent curves, declining quickly in the early stages of training and then gradually stabilized.

84.80%. The interesting thing is the best result we got only base 10% data from 1.6M tweets (local traversal for 160,000 tweets), instead of looping through the entire database (complete traversal). In this case, big data does not bring the ideal results with a more interpretable model, but huge time loss. The Figure 5.5 shows the development loss (as known as validation loss), train loss and accuracy during training, where the x axis denotes the steps which is iterations instead of epochs that usually used in Computer Vision field. Because the pre-trained model we use often has a deep network structure (12 layers for BERT base), usually only an extremely small learning rate (1e-6) is required to converge under single-digit epochs. In every step (mini-batch), calculating the loss and related evaluation metrics gives us an intuitive view of the model training process.

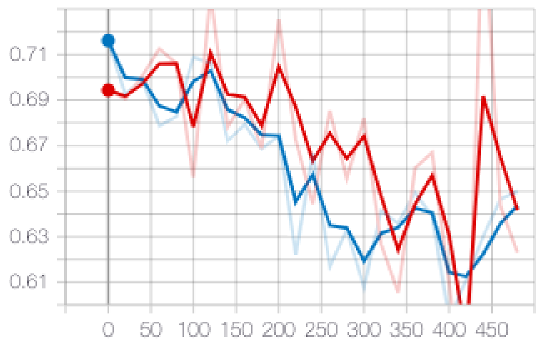
Compared to a public experiment result from Cai who use Deep Convolution Neural Network with GloVe Embeddings and also provide a baseline model result with CNN sentiment analyser, we summarise the results as Table 5.5. Our model performance fall short of the best (81.1% accuracy and 0.817 f1 score with bold).

Model	Accuracy	F1
Baseline	0.650	n/a
CNN+GloVe	0.811	**0.817**
10% data O-BERT	0.7820	0.7732
100% data O-BERT	0.7710	0.7670
Board Leader	**0.8480**	n/a

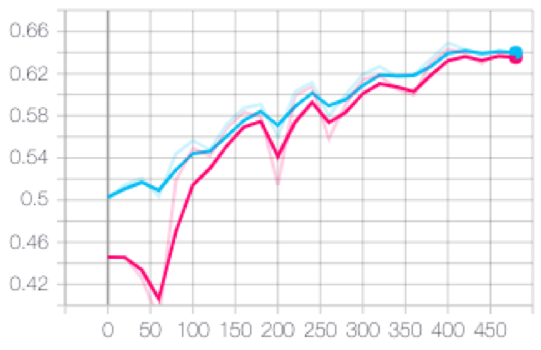
Table 5.5: Model performance on dev and test set.

Using sentiment140 dataset which have balanced data distribution, we also completed a experiment to show the equivalence of cross entropy and focal loss with focusing parameter $\gamma = 0$ as shown in Figure 5.6. We got an accuracy of 67% for both methods, indicating weighted sampling with cross entropy have the same expressive power as focal loss where weighted sampling is actually ineffective for balanced data.

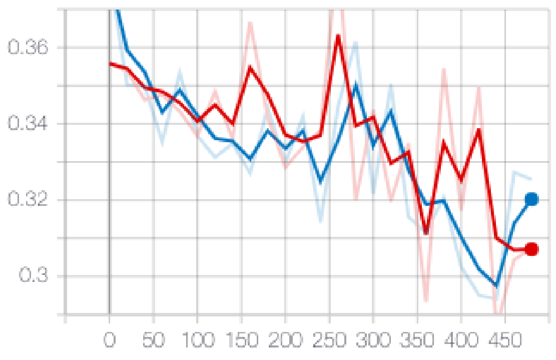
To evaluate the effect of focal loss in our sentiment classification, we artificially divide the sentiment into an unbalanced state. On a eighth-two dataset, we applied weighted sample and focal loss independently to deal with data imbalance. As seen in Figure 5.7, we conducted several controlled experiments and compared the model performance under the same experimental conditions, including using AdamW [30]



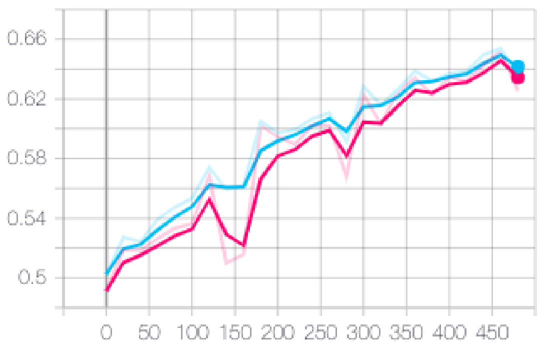
(a) loss of weighted sample



(b) results of weighted sample



(c) loss of focal loss with $\gamma = 0$



(d) results of focal loss with $\gamma = 0$

Figure 5.6: Equivalence of focal loss and cross entropy on weighted sample. use 1% data from sentiment140 (a balanced dataset with 8,000 samples for positive and negative category), two experiments yielded approximate results of 67% accuracy and f1 score of 0.66 under 5 epochs

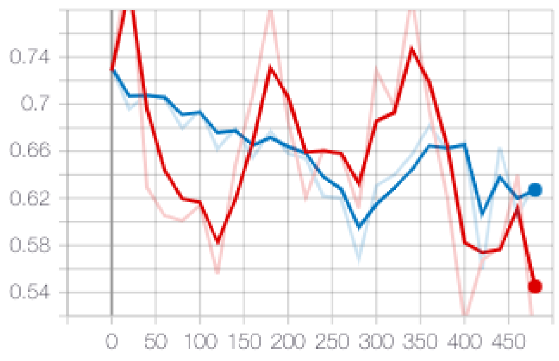
as optimizer with learning rate 1e-6 and batch size of 96. The above experimental conditions are consistent with the balanced dataset.

Meanwhile, we tested all these trained model on a balanced dataset with the same size, since we have no clue of specific emotional distribution of data on Twitter. As displayed in the Table 5.6, weighted sampling is better than CE on the balanced test set, which can be attributed to the balanced training set generated by weighted sampling is fit to the distribution of test set. Focal loss with higher focusing parameter γ shows excellent accuracy on the unbalanced dataset, while it does not migrate well to the balanced test set, reflecting less robustness. It can be seen that the data distribution has a significant influence on the model evaluation, which requires us to use the test set to simulate the actual application scenario. In the following experiments, we used balanced cross entropy as the loss function. There are different datasets and multiple application scenarios waiting for our pre-trained model, and it is demanding to select the hyper-parameter by cross-validation, while the focusing parameter γ was verified to be crucial in the experiment.

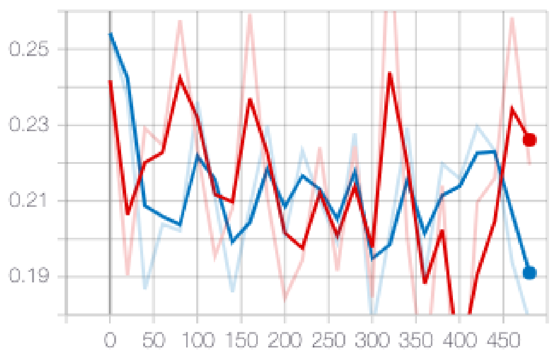
	Unbalanced test set		Balanced test set	
	F1	Acc	F1	Acc
Weighted sample+CE	0.6179	0.7243	0.6389	0.6463
FL ($\gamma = 0$)(α-balanced CE)	**0.6336**	0.7710	0.5946	0.6203
FL ($\gamma = 0.5$)	0.5269	0.5650	0.6262	0.6313
FL ($\gamma = 1$)	0.5673	0.6234	**0.6537**	**0.6571**
FL ($\gamma = 2$)	0.5892	0.6559	0.6204	0.6225
FL ($\gamma = 5$)	0.5633	**0.7921**	0.4522	0.5481

Table 5.6: Model verification on different test sets

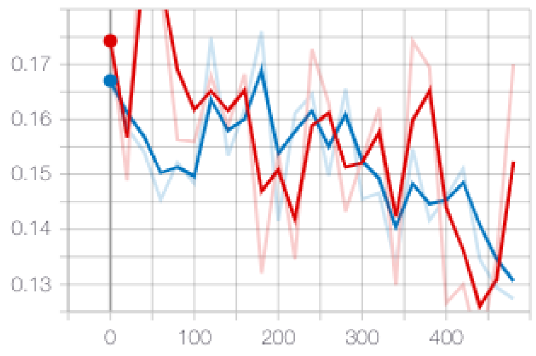
The reason of such bad results might be attributed to the collection methods. We imply divide the sentiment according to the emoticons that appear in the content. But, in the dataset provided, only texts are retained and the corresponding emoticons are erased. However, the emoticons precisely play the key to determine the emotional direction of the whole tweet.



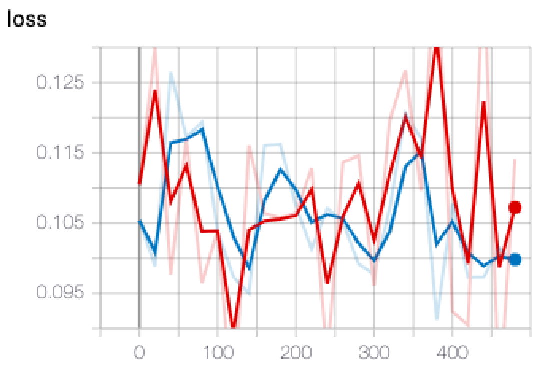
(a) Weighted Sample with CE



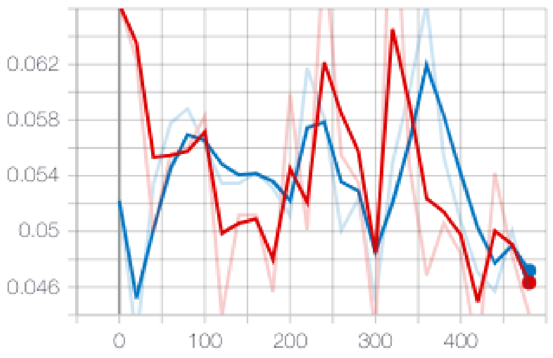
(b) Focal Loss with $\gamma = 0$



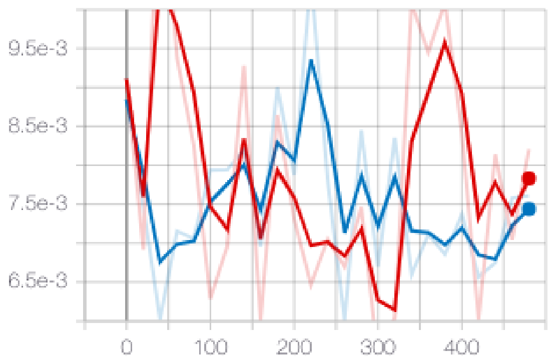
(c) Focal Loss with $\gamma = 0.5$



(d) Focal Loss with $\gamma = 1$



(e) Focal Loss with $\gamma = 2$



(f) Focal Loss with $\gamma = 5$

Figure 5.7: Performance comparison on CE and FL with different focusing parameter γ. Randomly extract 16,000 data from sentiment140 maintaining an imbalanced proportion of 80-20. Blue and red lins denotes training and development loss respectively, showing different drop rate although they might not convergence after merely 5 epochs.

Ternary Emoji Data set

It is the dataset we collect automatic based on the VADER (details in section 5.2.3) that contains rich non-text information. After performing the translation of the infrequent emojis (in section 5.3.1), the dataset still has text information worth exploring, which can be utilised as a training set for the plain text model, involved BERT-Origin, CNN-BERT and their frozen model.

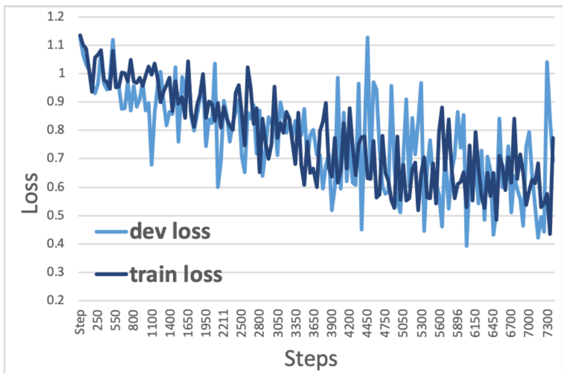
The Figure 5.8 and Table 5.7 show the performance of these models on our ternary emoji (3-emoji) dataset. For the two full training models (over 110M parameters) BERT origin and BERT CNN, we only run 10 epochs due to the pressure of time and training costs, taking 18 hours without convergence. In contrast, the frozen model which we fine-tuned the classify (CNN) leaving the parameters of multi-layer encoders fixed, has almost converged in half an hour (Figure 5.8(d)), showing mediocre results with accuracy of 67.30% and 68.34% for dev and test set. In view of the structural design principle, the ability to obtain and identify valid information by linear for the frozen model is theoretically tricky to outperform convolution linear, driving us to discard the experiment of linear classify on this dataset.

In addition, from a structural view, these two classifies receiving different inputs aimed at the same target of text classification. Linear classify accept the hidden state of the first token [CLS] while convolution classify use a more complex structure and input, that is the hidden state of the entire sequence that is a fine-tuned word embedding representation keeping same dimensions and probability distribution as the original input. However, complexity does not bring superiority to the model, leaving the same accuracy. The f1 score verifies the ideas that the non-semantic tokens [CLS] can be identified as a summary of the whole sentence.

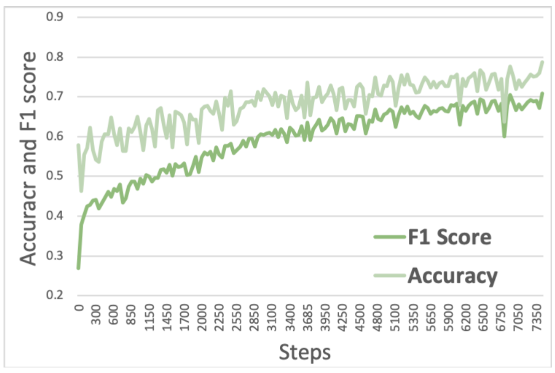
Model	Dev set		Test set		Time consumed
	Accuracy	F1	Accuracy	F1	
O-BERT	**0.7865**	**0.7087**	**0.8287**	**0.7589**	17h 54min 56sec
FCNN-BERT	0.6730	0.5263	0.6834	0.5762	Converge in 1h 20min
CNN-BERT	0.7787	0.7020	0.8246	0.7484	18h 44min 36sec

Table 5.7: Model performance on dev and test set.

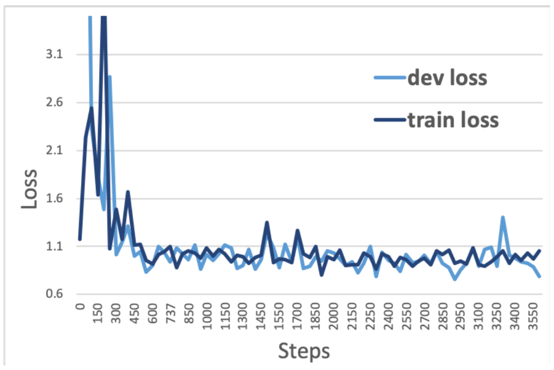
The above results reveal that it is unavoidable to fine-tune the entire model instead of merely using the original word embedding, even though it carries rich semantic



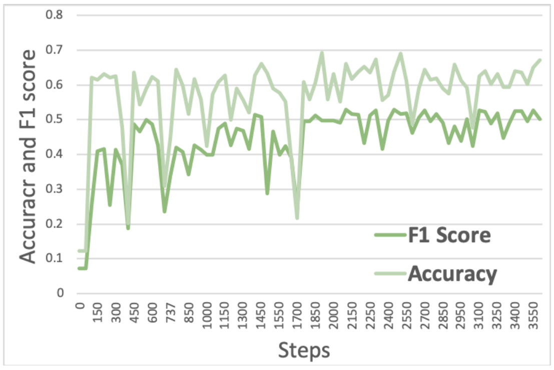
(a) Loss of Original BERT



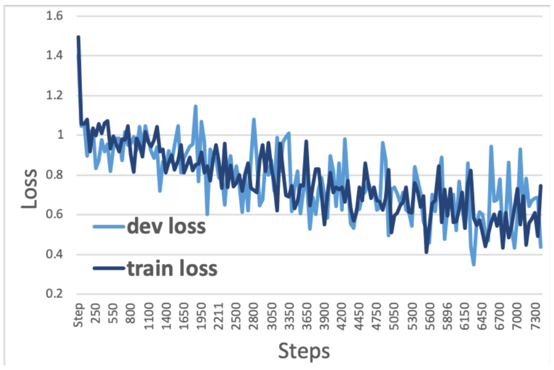
(b) Evaluation of Original BERT



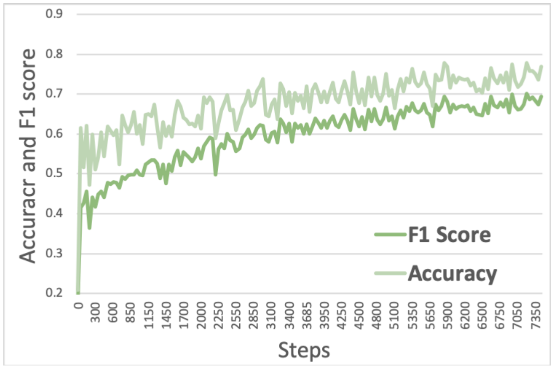
(c) Loss of FCNN-BERT



(d) Evaluation of FCNN-BERT



(e) Loss of CNN-BERT



(f) Evaluation of CNN-BERT

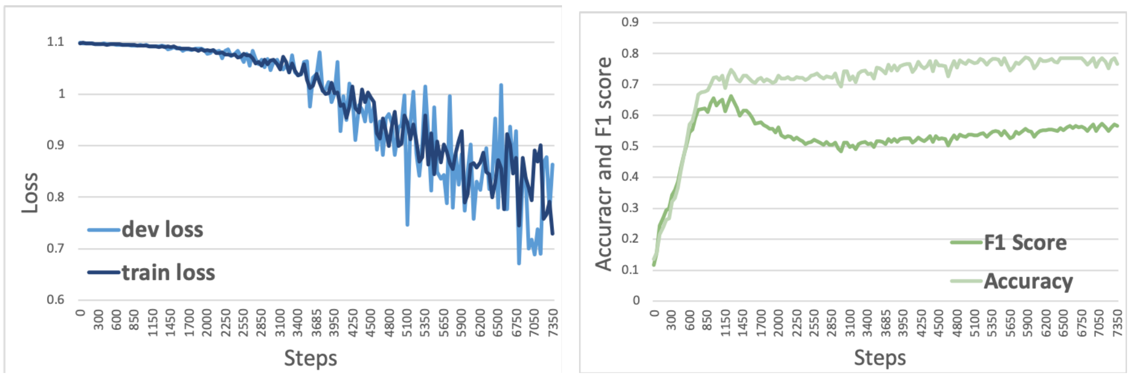
Figure 5.8: Text Model performance on 3-emoji Dataset. Linear classify and convolution classify show almost same expressiveness on plain-text dataset obtaining around 82% accuracy and 75% F1 score on test set, while frozen model shows satisfactory results.

information and prior knowledge. Combined with prior knowledge, model parameters can better match our downstream task under the well-classified data - sentiment analysis, verifying the original intention of introducing this pre-trained model before.

5.5.4 Emoji Attended Model

Fine-tuning from Original BERT

Under a well-trained O-BERT model, we discard the linear classify and construct a Emoji attended sub-encoder. Due to the lack of public data with Emoji, we use our collected 3-emoji dataset split into proportion 6:2:2 as train, development, test, to test the model performance. The primary purpose of the experiment is to verify that Emoji improves accuracy as an informational supplement.



(a) Loss of EA-BERT (b) Evaluation of EA-BERT

Figure 5.9: Training results for EA-BERT model

We use a well-trained model on a the same dataset focusing on the plain-text for fine-tuning, as seen in Figure 5.9 and Table 5.8. Sadly, we got no more improvement with a similar results as in Table. Experiments show that the addition of Emoji is not helpful for emotional judgement, which is not in line with common sense. We deeply study the weight coefficient of text classifier and Emoji classifier, obtaining a value of 0.5 and 0.5 through a softmax function, which conforms that the stacking of linear layers does not affect the results of the model, even if the weight coefficient is changed to be set artificially.

Interestingly, at the beginning of the training, the loss did not decrease, but accuracy and F1 scores continued to rise. This phenomenon may be since the model has been adequately trained in plain text and can quickly adjust the parameters related to emoticons by fusing previous features. However, it is precisely this reason that causes F1 score to decline slowly and cannot be corrected later. In a word, adding Emoji attended parameters to original bert model makes the model unbalanced, which restricts the emoji ability to connect text and capture contextual semantic. It can be expected that training starting from zero could have better results, because of the cost of training, it will be left to work in the future.

	Dev set		Test set	
	Accuracy	F1	Accuracy	F1
BERT-Origin	0.7865	7087	0.8287	0.7589
EA-BERT	0.7882	0.6629	0.8193	0.5963
FEA-BERT	0.8383	0.7656	0.8763	0.8080

Table 5.8: Comparison of Emoji attended model and its feeding vision. Not done yet for incorrect operation on training server.

5.5.5 Training on Emoji-Feeded Dataset

Considering Emoji, which simulates facial emotions, has obvious emotional tendencies. As the data presented by Table 5.1, different Emoji occurs in different language environments with different frequencies, while positive expressions account for the vast majority.

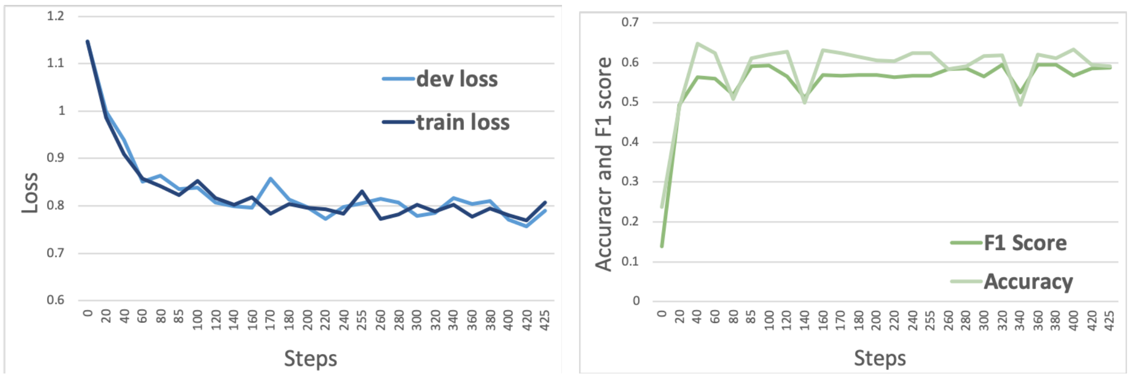
After studying the datasets, we found that in our 3-emoji dataset, there are only 1,300 labelled data with Emoji (after translating) in totally 110,659, which demonstrates a marked difference between the number of words and Emoji.

In order to make up for the shortcomings of the above experiments, we have given Emoji a lot of prior knowledge on the basis of retraining, that is, using an artificial dataset to pre-train Emoji embedding separately. It is no wonder that the EA-BERT model in the above experiments only obtains similar accuracy compared to O-BERT model. To make up for the shortcomings of the above experiments, we have given Emoji amount of prior knowledge, that is, using an artificial data pre-train Emoji embedding independently, feeding with sensual meaning.

Based on the Occurrences of top 55 Emojis and two usual emoctions (i.e. ":)" and

":(")", we construct a dataset containing 126,110 automatic labelled data. To be specific, for (face with tears of joy), we simulate its probability distribution in the real scene according to the statistical data provided [38], i.e. 3614 negative, 4163 neutral and 6845 positive. Furthermore, the emergence proportion of different Emoji is also considered into the design of feeding data, while for happy emoticon and sad emoticon, we extracted 10,000 lines of data each and identified them as having certain sentiment, with all blank text content .

Batch size 1024 is used instead of 96 for above all experiments, because we set the max length of text as 4 focusing the reinitialization of emoji embedding and the leave the fine-tuning of attention parameter for the rest experiment which is absent for the reason of misoperation on the server.



(a) Loss of Emoji feeding (b) Evaluation of Emoji feeding

Figure 5.10: Training on Emoji-feeded dataset. The training process has undergone several adjustments of learning rate, so only the loss curve at the end stage is selected.

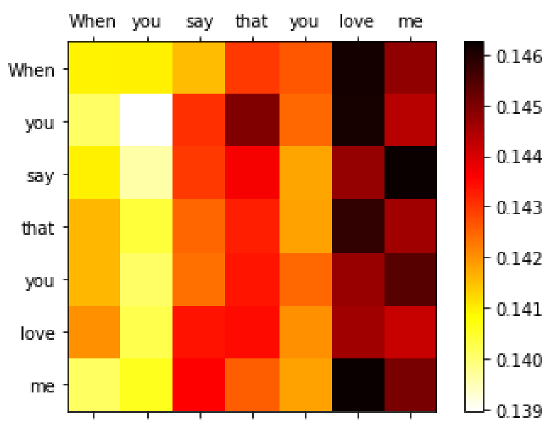
As seen in Figure 5.10, getting an accuracy of 66% on this feeding data, test of the next phase of the is highly anticipated which we can call this Emoji-feeded model as Feeded Emoji-Attended BERT (FEA-BERT) with the same structure with EA-BERT. This seemingly low accuracy rate (65%) actually reflects the true usage of Emoji on Twitter. After all, we can not rely solely on Emoji to determine the emotional judgement of its tweet but explore the semantic interaction with the text.

For the results of the FEA-BERT model, due to the misoperation at the end of the experiment, all related records are erased. However, it is a burdensome training task that takes about 20 hours on four GPUs. We are doing our best to recreate the previous work and these crucial experiments.

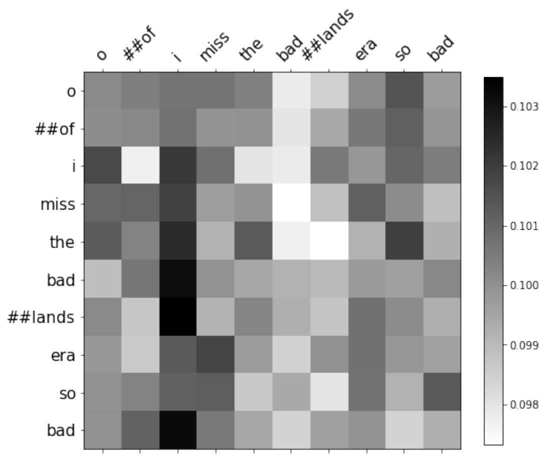
Chapter 6

Results and Discuss

6.1 Attention Visualization



(a) Positive word “love”



(b) Negative word “bad”

Figure 6.1: Visualization of attention mechanism

Based on the attention mechanism, our proposed models are able to grab key information and semantics between words. AS Figure 6.1 depicted, attention weights generated by the 12-layer network tend to automatically focus on strong emotional words. The original intention of attention, to explore the dependency between words, will not be completely erased because of our downstream tasks, such as the word love and its object is still highlighted. (see in sub-figure 6.1(a)). In the right sub-figure 6.1(b), the word "bad" conveying negative emotions is also automatically captured, showing bright black.

6.2 Web Application Interface

A responsive web application based on bootstrap[1] has been constructed to dynamically response and re-size each component according to different user behaviour and access devices including person computing, Android or IOS devices having with screen size. we refer to some open-source web interface [2], designed a delightful interface of blue and white as the main colour.

We entitled the web application as "Test Before Tweet" aimed at testing whether the emotion presented by the tweet conforms to the intention. In addition, the user can know the objective emotional feeling from other social media users about the tweet to be released. Served as a free public relationship (PR) tool, "Test Before Tweet" can make users feel the impact of emotional expression, which is closely attached with social media now.

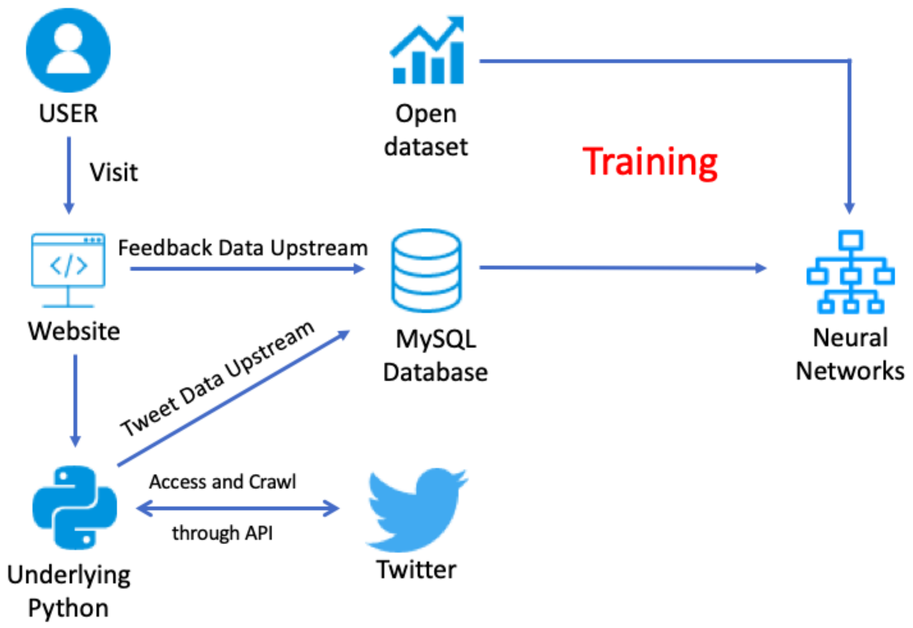


Figure 6.2: Illustration of web application work flow

[1]https://getbootstrap.com/
[2]`https://github.com/BlackrockDigital/startbootstrap-sb-admin-2`

It is a python-PHP based web application, and such a cumbersome design does violate the original intention. Same as our training machine, this web application was deployed at Microsoft Azure, while it is still open to the public and can be continuously accessed[3]. Figure 6.2 depicts the overall work flow of our web application, connection with MYSQL database to store well-labelled tweets.

The web application itself was built on PHP and JavaScript. JavaScript was used to complete the front-end delivery work and coordinate with HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) to display the web interface stably, supporting scrolling, directory and other interactive functions. The job of data acceptance and delivery is done by Ajax (Asynchronous JavaScript And XML) which allows web pages to respond more quickly by exchanging data with the web server behind the stage instead of reloading the entire web page whenever the user sends a request. PHP and Nginx [4] played an important role for launch. As an efficient web service proxy, Nginx can support more concurrent connection access with fewer resources. Php was used as a background language connecting to the local database, which ensures that the manually labelled data can be stored properly. Regarding python mentioned in the work flow, it is necessary for the usage of VADER which is implemented in python, of course, our proposed model is also python-based.

During the development, several issues were settled as below:

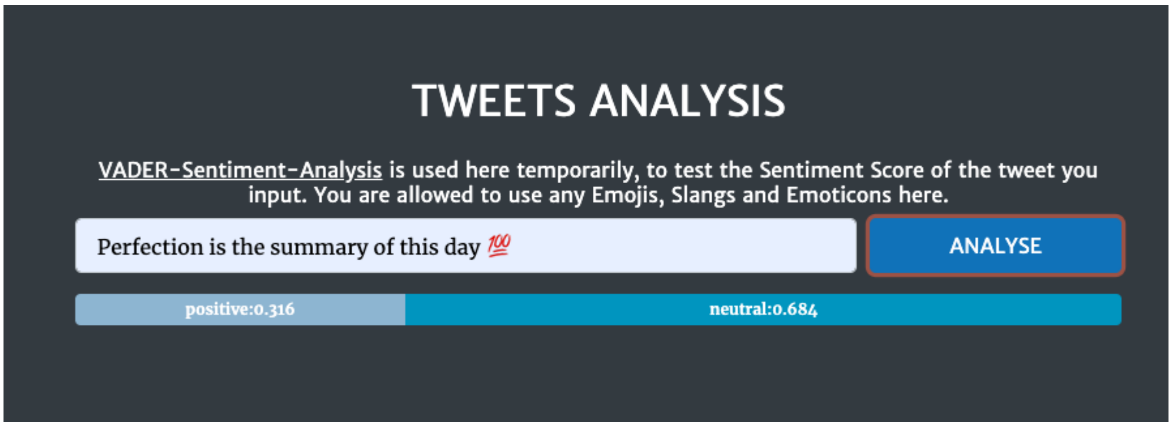
- Multithreading Issues that allows numerous people to access web application simultaneously. We use a thread pool to allocate access requests that can be executed concurrently.
- Database downtime caused by long idle time (8 hours).
- Crack the relationship between the tweet id generated by snowflake and the time series, satisfying the demand to crawl Tweets according to time. It also offers a remedy to pseudo-random search algorithm of twitter API.

Offering a variety of functions, this web application was formed by three main modules: Annotation, Analysis and Tracking. Annotation is a step collecting data from online volunteers allowing the model to be trained in a large dataset (Figure 5.1). The Analysis module receives an input tweet and returns an output of sentiment

[3]`http://testbeforetweet.uksouth.cloudapp.azure.com`

[4]`http://nginx.org/en/`

index (Figure 6.3) where VADER was used before the model fixed. Tracking allows the visitors of the web application to search a user, supporting fuzzy input of screen name, and tracks twitter usage, besides generate a line chart to auxiliary illustration (Figure 6.4 and Figure 6.5).


TWEETS ANALYSIS

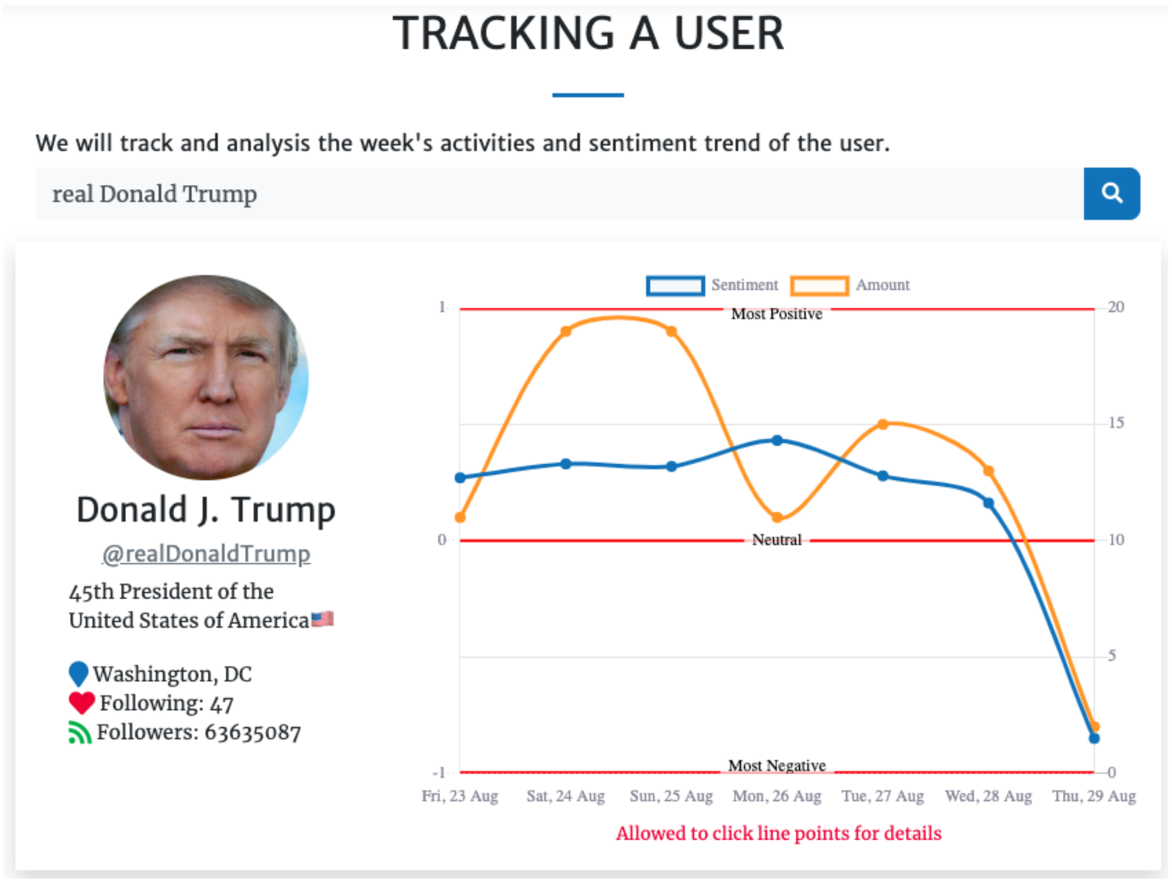
VADER-Sentiment-Analysis is used here temporarily, to test the Sentiment Score of the tweet you input. You are allowed to use any Emojis, Slangs and Emoticons here.

Perfection is the summary of this day 💯

Figure 6.3: Tweet analysis module. Given a input sentence, it returns a sentiment results composed of three categories of emotions with a sum of 1.

Figure 6.3 shows a beautiful interface with a search box allowing the user to input a sentence to be tested. English is the only language we accept for now, but Emoji and emoticon are available. This part has been built before the data collection and model training, so the VADER sentiment analysis is used temporarily and will be adjusted later.

The module of tracking is illustrated in Figure 6.5(b), Analysing Twitter usage during the last 7 days with a full description of the search user and a line graph showing the number and average sentiment for each day. The visitors of the web application are also allowed to click the line point for daily details as shown in Figure **??**, and chart is placed below the avatar when receiving the access request from a mobile device, with an aim to adapting to the natural browsing habits.


TRACKING A USER

We will track and analysis the week's activities and sentiment trend of the user.

real Donald Trump

Donald J. Trump

@realDonaldTrump

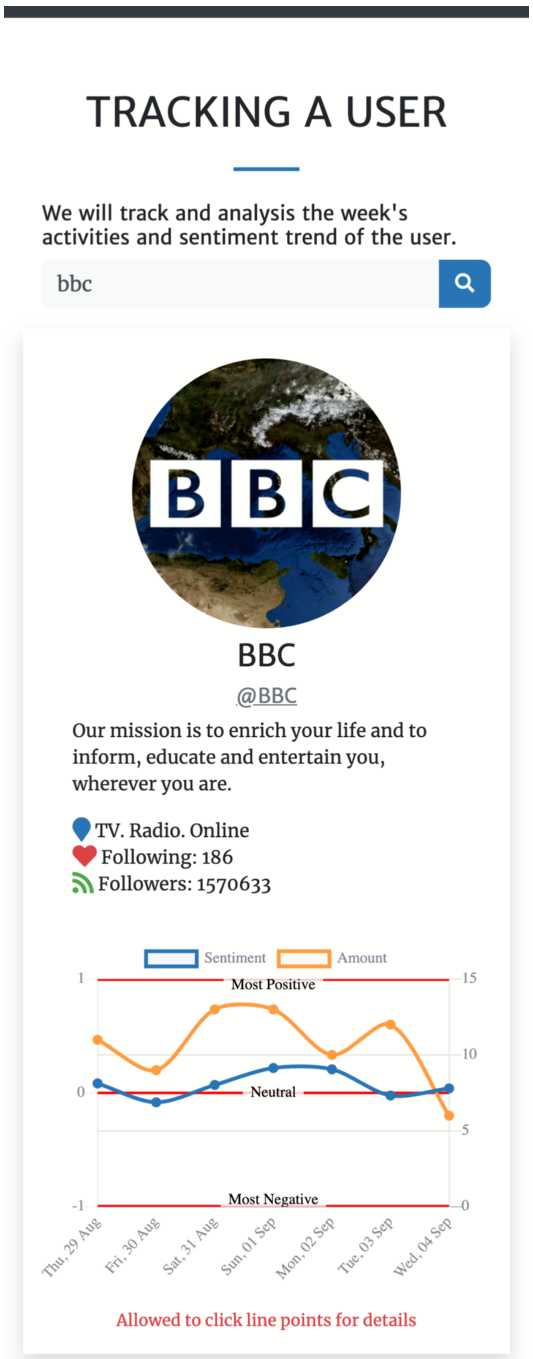
45th President of the United States of America🇺🇸

Washington, DC

Following: 47

Followers: 63635087

Figure 6.4: Tracking Module for Searched User. On the left side of the figure is the user details, including screen name, description, location, and quantity of followings and followers. While, the line graph on the right shows the user's Twitter usage in the last week, the orange and blue line indicate the amount and the sentiment respectively.



TRACKING A USER

We will track and analysis the week's activities and sentiment trend of the user.

bbc

BBC

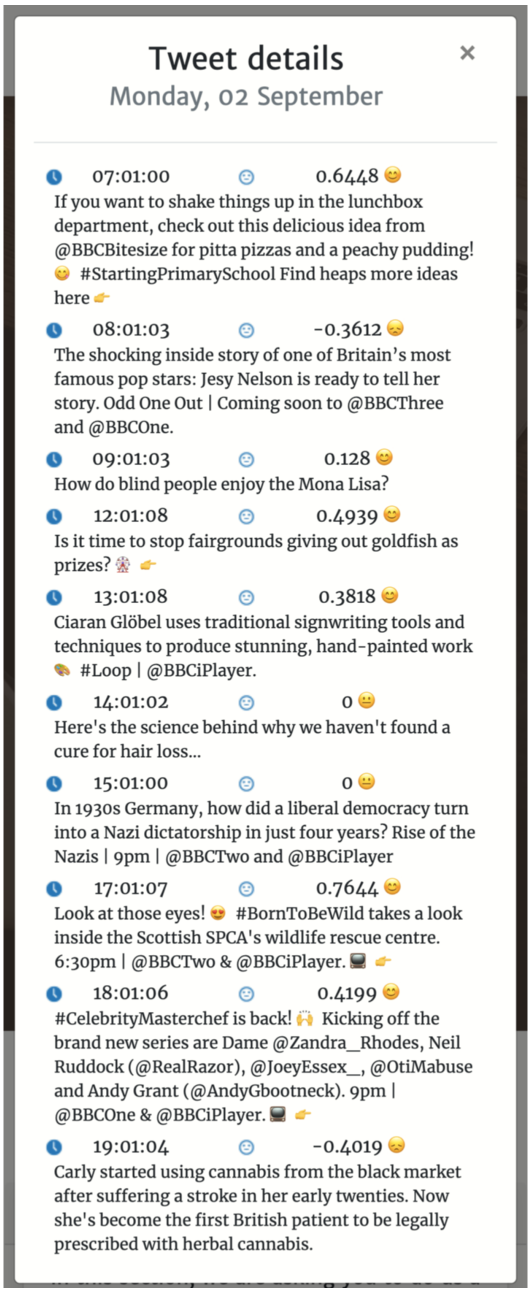
@BBC

Our mission is to enrich your life and to inform, educate and entertain you, wherever you are.

TV. Radio. Online
Following: 186
Followers: 1570633

Allowed to click line points for details

(a) Module view from mobile phone



Tweet details
Monday, 02 September

07:01:00 0.6448
If you want to shake things up in the lunchbox department, check out this delicious idea from @BBCBitesize for pitta pizzas and a peachy pudding! #StartingPrimarySchool Find heaps more ideas here

08:01:03 -0.3612
The shocking inside story of one of Britain's most famous pop stars: Jesy Nelson is ready to tell her story. Odd One Out | Coming soon to @BBCThree and @BBCOne.

09:01:03 0.128
How do blind people enjoy the Mona Lisa?

12:01:08 0.4939
Is it time to stop fairgrounds giving out goldfish as prizes?

13:01:08 0.3818
Ciaran Glöbel uses traditional signwriting tools and techniques to produce stunning, hand-painted work #Loop | @BBCiPlayer.

14:01:02 0
Here's the science behind why we haven't found a cure for hair loss...

15:01:00 0
In 1930s Germany, how did a liberal democracy turn into a Nazi dictatorship in just four years? Rise of the Nazis | 9pm | @BBCTwo and @BBCiPlayer

17:01:07 0.7644
Look at those eyes! #BornToBeWild takes a look inside the Scottish SPCA's wildlife rescue centre. 6:30pm | @BBCTwo & @BBCiPlayer.

18:01:06 0.4199
#CelebrityMasterchef is back! Kicking off the brand new series are Dame @Zandra_Rhodes, Neil Ruddock (@RealRazor), @JoeyEssex_, @OtiMabuse and Andy Grant (@AndyGbootneck). 9pm | @BBCOne & @BBCiPlayer.

19:01:04 -0.4019
Carly started using cannabis from the black market after suffering a stroke in her early twenties. Now she's become the first British patient to be legally prescribed with herbal cannabis.

(b) Details for a specific day

Figure 6.5: Tracking tweet of @BBC (collect from 4th September midnight to 14 pm). Accessed by clicking line points, this sub module shows multiple tweets with corresponding sentiment score.

Chapter 7

Conclusion and Future Work

7.1 Contribution

In this thesis, a basic model is constructed for tweet sentiment analysis. In addition, the application of natural language processing and deep learning, which truly benefit users in daily life and social media networks, is developed. The main contributions can be concluded as below:

- A page-friendly website was constructed to support our need for manual data annotation. Using the wisdom to the crowed principle, emotional judgements were collected on tweet that is labelled by online annotators. Several validation methods were proposed to filter the injection of messy information.

- On the basis of manual data collection, using VADER as data supplement and automatic label tool, a well-distributed dataset is obtained after data cleaning with contraction mapping and graphic characters translation. The original data comes from automated crawling of daily Twitter through a public API. In addition, the validity and reliability of VADER are verified by manually labelled data.

- In this thesis, we proposed two basic models that take 12-layer transformer encoder as feature extractor, linear and CNN as the classifier, namely Original BERT and CNN BERT respectively, making full use of BERT pre-trained model. On multiple datasets, the model performance on pure text is measured by using

F1 score and attention as evaluation metrics. Experiments show that although CNN classifier accepts more complex inputs and adopts more complex structure, it does not benefit the improvement of the results.

- we propose an emoji attended model, based on a well-trained text classification encoder with the multi-head self-attention mechanism. Emoji is regarded as additional information and attended to the text to explore their interaction based on the original model, but the desired effect is not satisfied.

- Based on the shortcomings of the above experiments, it is found that the model is unbalanced because the effective pre-trained text occupies a dominant position in the attended mixed vectors, and can not show excellent results. To this end, we combine the emotional prior knowledge of Emoji to generic. The Emoji embedding is gradually strengthened under the moistening of the dataset. The obtained result with 65% accuracy is basically in line with expectations and can be used in a subsequent experiment.

7.2 Future Work

After years of education, mankind has formed a basic world outlook, including all aspects of life, such as culture, nature, society and spirit, and of course, language. They are all prior knowledge that is conducive to grasp the key information but is unreasonable to forward the information for the machine. To enable the machines to understand human language, they can only depend on a large dataset or a model with strong capabilities of semantic extracting, which we lack in this thesis.

In consideration of many shortcomings and ill area that need to improve in this thesis, future works of sentiment analysis can be carried out by the following aspects:

- Both sentiment 140 and ternary Emoji are generated based on the automatic annotation, where the former one depends on the emoticon, and the latter one is based on VADER. Data resources are the key to the application of sentiment. More data resources are important to the sentiment analysis of social media, while the internet is inundated with emoji usage case but lacks effective annotation. In view of the preciousness of manual labelling data, only 337 reliable data were available in our case, the proposal of automatic annotation method is an important research direction in the future.

- In the above experiments, one important missing point is to complement the Emoji attended model by using Emoji emotion that combined with the attention mechanism. It's a good start that the embedding vectors of emoji are strengthened with a simulated probability distribution, but no valid data is available because of the server incorrect operation.

- In the experimental process, there is a lack of comparison between the proposed model and baseline models such as LSTM or traditional machine learning algorithm SVM on ternary Emoji dataset, which is inevitably biased on the conclusion and depends on self-comparison.

- During the training process, each of the 12-level encoders produces an attention weight. While we only show the attended results of the last level in the results section, the semantic relations, which have obtained under unsupervised pre-training, are weakened. Our proposed model focuses more on keyword locking, which unavoidably exposes some possible causes. For grammatical problems, dealing with double negative sentences will confuse the model. The focus of later work can be focused on the extraction of attentions in the middle layer.

Bibliography

[1] Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer Normalization. *arXiv.org*, July 2016. pages 17

[2] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv.org*, September 2014. pages 13, 14

[3] Michael Cai. Sentiment analysis of tweets using deep neural architectures. Stanford CS224N Natural Language Processing Project Report, 2019 URL `http://web.stanford.edu/class/cs224n/reports/custom/15786247.pdf`. pages 55

[4] Yuxiao Chen, Jianbo Yuan, Quanzeng You, and Jiebo Luo. Twitter sentiment analysis via bi-sense emoji embedding and attention-based lstm. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 117–125. ACM, 2018. pages 37, 38, 46, 48

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014. pages 12, 13

[6] 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. ACM, 2008. pages 10

[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from

scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011. pages 2, 7

[8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Association for Computational Linguistics (ACL)*, 2019. pages 21, 26

[9] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 241–249. Association for Computational Linguistics, 2010. pages 37

[10] 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 Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. pages 21, 23, 24, 25, 26, 31, 32, 51

[11] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017. pages 9

[12] Tina Ganster, Sabrina C Eimler, and Nicole C Krämer. Same same but different!? the differential influence of smilies and emoticons on person perception. *Cyberpsychology, Behavior, and Social Networking*, 15(4):226–230, 2012. pages 37

[13] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009. pages 6, 41, 48, 53

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. pages 10, 12

[15] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017. pages 50

[16] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954. pages 8

[17] Vasileios Hatzivassiloglou and Kathleen R McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics, 1997. pages 6

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. pages 17, 18, 34

[19] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415, 2017. pages 33

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`. pages 12

[21] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. pages 9

[22] W John Hutchins. Machine translation over fifty years. *Histoire epistémologie langage*, 23(1):7–31, 2001. pages 2

[23] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014. pages 3, 6, 27

[24] Dan Jurafsky and James H Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J. : Pearson Prentice Hall, second edition, 2009. pages 2

[25] Kim, Yoon. Convolutional Neural Networks for Sentence Classification. *arXiv.org*, cs.CL:1–6, September 2014. pages 10, 36

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, 2015. pages 50

[27] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. pages 49

[28] Bing Liu. Sentiment analysis and subjectivity. In N. Indurkhya and F. J. Damerau, editors, *Handbook of natural language processing*, pages 627–666. CRC Press, 2 edition, 2010. pages 3, 5

[29] Chuan Liu, Wenyong Wang, Meng Wang, Fengmao Lv, and Martin Konan. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowledge-Based Systems*, 116:58–73, 2017. pages 41

[30] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2018. pages 50, 55

[31] James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009. pages 9

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. pages 7, 8, 23

[33] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013. pages 7, 36

[34] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009. pages 9

[35] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004. pages 6

[36] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999. pages 6

[37] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000. pages 6

[38] Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. Sentiment of emojis. *PloS one*, 10(12):e0144296, 2015. pages 42, 63

[39] Christopher Olah. Understanding lstm networks – colah's blog, 2019. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. pages 12

[40] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning in natural language processing. *arXiv preprint arXiv:1807.10854*, 2018. pages 10, 12

[41] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *The International Conference on Language Resources and Evaluation*, volume 10, pages 1320–1326, 2010. pages 3, 6

[42] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002. pages 3, 6

[43] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008. pages 1, 2, 3

[44] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. pages 51

[45] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. pages 7, 10, 23, 36

[46] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, 2018. pages 22

[47] Rosalind W Picard. Affective computing mit press. *Cambridge, Massachsusetts*, 1997. pages 2

[48] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009. pages 3, 5, 6

[49] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018. pages 21, 22, 24

[50] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. pages 12

[51] Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. Multi-perspective question answering using the opqa corpus. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 923–930. Association for Computational Linguistics, 2005. pages 6

[52] James Surowiecki. *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations.* Anchor, 2004. pages 27

[53] 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. pages 13

[54] Wilson L Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433, 1953. pages 25

[55] Rahul Tejwani. Sentiment analysis: A survey. *ArXiv*, abs/1405.2584, 2014. pages 2, 3

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`. pages 16, 17, 19, 20, 32

[57] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016. pages 14

[58] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. pages 30

[59] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. pages 14

[60] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv.org*, June 2019. pages 22, 23, 26, 27

[61] R Sandra Yuwana, Endang Suryawati, and Hilman F Pardede. On empirical evaluation of deep architectures for indonesian pos tagging problem. In *2018 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 204–208. IEEE, 2018. pages 2

[62] Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1528–1531. ACM, 2012. pages 37

[63] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015. pages 32

Appendix A

Ethics Checklist

Table A.1: Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?	✓	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	✓	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓

Table A.2: Ethics Checklist

	Yes	No
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing datasets?		√
Section 5: ANIMALS		
Does your project involve animals?		√
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		√
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		√
Could the situation in the country put the individuals taking part in the project at risk?		√
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		√
Does your project deal with endangered fauna and/or flora /protected areas?		√
Does your project involve the use of elements that may cause harm to humans, including project staff?		√
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		√
Section 8: DUAL USE		
Does your project have the potential for military applications?		√
Does your project have an exclusive civilian application focus?		√
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		√
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		√

Table A.3: Ethics Checklist

	Yes	No
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		√
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		√
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		√
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		√
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		√
Will your project use or produce goods or information for which there are data protection, or other legal implications?		√
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		√