Deep-Learning based Nature Disasters Detection From Social Media

Pangolin:
Yinglu Jia
Bin Ma
Beichen Zhang

May 1, 2021

Contents

Abstract

The detection of natural disasters draws much attention in recent years, as the information sources have been largely expanded by technology development. Twitter, as one of the largest international social media platforms, could be a good data source. In the final project, we employed text-based tweet data to identify whether the tweets were related or unrelated to natural disasters. We trained and tested four types of models: long short-term memory (LSTM), bidirectional long short-term memory (BiLSTM), gated recurrent unit (GRU), and bidirectional encoder representation from Transformers (BERT). The differences between pre-trained and self-trained word-embedding models were explored by comparing the LSTM-based models with the different word-embedding models. Hyperparameter tuning and various regularization methods were also applied in the project. Overall, the BiLSTM with a pre-trained word embedding model and BERT had a similar performance that was better than LSTM and GRU. And adding a pre-trained word embedding model help improve models' performance. The best model was a BERT model that achieved a test accuracy of 81.3%. However, all kind of models had a significant overfitting problem. Although we tested various regularization methods, including dropout layers, L1 and L2 norms, global pooling, layer normalization, and batch normalization, the problem was not significantly addressed. We will focus on overcoming the overfitting and improving hyperparameter tuning and text processing steps to achieve a higher prediction accuracy in future work.

Chapter 1

Milestone 1: Project Ideas

1.1 Introduction

The impact of natural disasters has been growing by the decade. From the 1960s to 2000 the economic losses caused by disasters increased five times. This is aroused together by ongoing climate change and rising human activities which affect hydrological cycle both in spatial and temporal aspects. As a result, the hydrological cycle changes tend to affect the intensity and frequency of extreme weather and climate conditions, which lead to increasing natural disasters. Based on the Intergovernmental Panel report (IPCC) on climate change's impacts on the natural environment, 20-year extreme temperature appears to become a 2-year extreme, and 20-year extreme precipitation is likely to change to a 5-year extreme event in the 21st century. Besides, due to the expected increasing frequency and intensity and the changing spatial and temporal patterns of the precipitation and temperature, the floods and droughts, which are two common natural disasters, would probably intensify and be frequent in some catchments or climate regions [1]. Therefore, in the foreseeable future, addressing the problems of natural disasters caused by climate and weather extremes will be one of the most urgent tasks for human survival.

While natural hazards cannot be prevented from occurring, risk management is significantly important to reduce the major damages on society, economy, and ecosystem. Based on a study by the National Institute of Building Sciences (NIBS) in 2016, every 1 dollar in the natural hazard mitigation grants can save 6 dollar in future disasters costs [2], which revealed a significant contribution of mitigation in disaster management. Figure 1.1 shows a typical natural disaster risk management cycle [3]. In the risk management cycle, preparedness and mitigation are two crucial components that help the society proactive to a natural disaster in future and reduce its the negative impacts. For many natural disasters, the development of a mutual monitoring and early warning system could directly improve the response time and help policymakers review

and refine related mitigation plans and strategies. Hallegatte pointed out that the hydro-meteorological early warning system in world could rough reduce at least 0.3 billion (USD) damages from natural disasters [4].

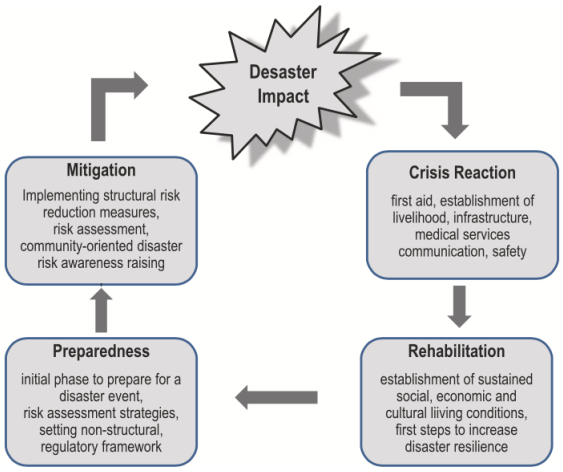

Figure 1.1: Natural disaster risk management cycle [3].

Entering in the 21st century, with the rapid development of the Internet, online mass media and social media makes it become progressively important to provide instant and valuable information in a natural disaster event. As a participatory and collaborative structure, social media has the functions of acquiring disaster awareness information, supporting self-organized peer-to-peer help activities and enabling the disaster management agencies to hear from the public[5]. Because of appealing advantages of obtaining information from social media, the number of environmental studies that used social media as their data sources increases since 2011 [6], For example, a study in Japan found that an earthquake warning system based on Twitter could detect earthquakes earlier than the Japanese Meteorological Agency[7].

However, because of the size of social media's data and its real-time characteristic, it has to be automatically processed The information obtained from internet. Hence, in the final project of this course, our group proposes two ideas of utilizing the techniques from natural language processing (NLP) in deep learning (DL) to conduct the text classification (TC) based on the data from social media and online media reports. The first idea is to recognize whether the tweet includes any information about natural disasters, which is the very first task is to monitoring general natural disasters. While the second one is to classify online raw reports into multi-labels of drought impacts. This work can help to save a great deal of time and labor effort.

1.2 Project Idea 1: Natural Language Processing (NLP) with Disaster Tweets

1.2.1 Background

Twitter is an American micro-blogging and social networking service on which users post and interact with messages. As smart phone goes more and more popular, twitter has more than 330 million active users, and hundreds of millions of tweets are posted everyday. As a micro-blogging, Tweet publishes information faster than newspapers and blogging, making it a nearly real-time media. These properties make Twitter a promising channel to efficiently search for useful information for disaster responds. In thus, many humanitarian agencies, like disaster relief team, are interested in monitoring Twitter. The twitter data could provide these organizations with real-time information on urgent needs of affected people, situation awareness of disaster impact, infrastructure damage et.al. In this way, they can save lives and provide aid effectively.

The first step of monitoring disaster reports on Tweet requires recognising the disaster-related information buried in the tons of tweets. Manually generating reports and summaries of tweets by volunteers and other support personnel is not practical considering the huge volume of data produced within a short time interval during a disaster. Therefore, it is necessary to pragmatically classify whether the tweet is disaster-related or not. The difficulty of this classification is that human language is not straight-forwardly understandable by machine, not to say various expression and metaphors used by human, making it harder to determine whether a person's words are actually announcing a disaster. For example, a tweet as "Look at the sky last night. It was ablaze", is clear to human, but machine may relate it to wildfire.

1.2.2 Data and Approaches

In application, there are various ways to solve this binary TC problem. Before the late 80s, the most popular approach to TC was a knowledge engineering (KE) in real application[8]. This method depended on a rule system manually defined by domain exports to encode their knowledge of how to classify these documents. The native disadvantage of this method comes from the intervention of human, which requires to formalize written rules. And it is not general to different domains. So various experts are needed to set rules for their own domains. As a result, it is expensive to implement and maintain rule systems.

In the 90s, the automated TC gains continuously increasing interests, which is a part of NLP strategies. This approach is mainly based on machine learn-

ing techniques which automatically builds a classifier by learning a set of pre-classified documents. The property of generality of this method means it is transferable to different domains, leading to a considerable savings of expert labor. The accuracy of this method is also comparable to human experts. In detail, a NLP model falls into 4 different parts: preparation, document representation, classifier construction and classifier evaluation (Fig. 1.2). preparation is usually a pipeline combined with segmentation [9], tokenization[10], and part-of-speech (POS) tagging[11]. For representaion, there are various methods to choose. For example one-hot encode[12], count vector[13] and more comprehensive ones like word embedding[14]. There are also a wide variety of classifiers constructed and applied to solve TC in both theory and practice aspects, including decision trees (DT)[15][16], neural networks (NN) [17][18], naive Bayes (NB)[19], support vector machines (SVM)[20], and k nearest neighbor(kNN)[21]. For classification problem, we can evaluate classifiers by comparing their confusion matrix accuracy,receiver operating characteristic (ROC) curve and area under the ROC curve (AUC) [22].

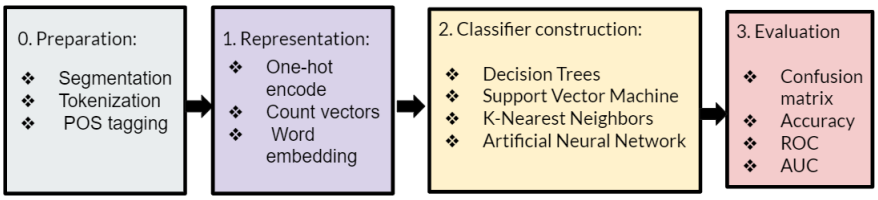

Figure 1.2: General methods of NLP

Here, We propose to build a NLP-based model to predicts which Tweets are about real disasters and which ones are not. we will use a tweets data set got from Kaggle[23] and apply different algorithm on this TC problem. As shown in Figure 1.3, the data includes training data and test data. There are 7503 entries in the training data. So, we will use that as both our training data and validation data. Besides the target feature which is the label, the data includes: id, key word, location and text. The text feature will be the text-based content of the tweets.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

Figure 1.3: The data structure of NLP with disaster tweets

1.3 Project Idea 2: Classification of Drought Impact based on Media Reports from Drought Impact Reporter (DIR)

1.3.1 Background

Drought is one of the most costly natural disasters globally because of its broad impacts [24]. Ongoing climate change is inclined to increase the frequency and intensity of drought by raising extreme variabilities over space and time in the hydrological cycle[25][26]. However, compared to other natural disasters, such as floods and wildfire, drought impacts often lack structural and visible existence. Thus, drought impacts on different socio-economic aspects could be either tangible or intangible, direct or indirect [27][28]. For example, some drought impacts on agriculture, such as crop failure, can be easily counted. However, some other impacts, such as groundwater abstraction and its impacts on local watersheds and ecosystems, are much harder to be assessed. Additionally, based on the propagation of a drought event, its impacts could last for weeks to even years [24]. When a long-term drought event is ongoing, the drought could negatively impact many other social sectors, such as energy, tourism, and recreation, which won't be significantly affected by a short-term and rapid drought. The human-modified changes in climate and geographical environment make the drought-related impacts even more complicated. Identifying various drought impacts based on hydro-meteorological indicators would be more challenging.

Starting from 2005, the National Drought Mitigation Center (NDMC) developed a drought-information online database - Drought Impact Reporter (DIR) to fill the existing drought indices gap. Ones can access the DIR on the NDMC's website (Figure 1.4). DIR collects data from multiple resources that include but are not limited to media, users' observation reports, national weather station, Collaborative Community Rain Hail and Snow Network (CoCoRaHS). And online media takes a significant proportion of the data. The impact data are then classified into nine categories: agriculture, business & industry, energy, fire, plants & wildlife, relief, response & restrictions, society & public health, tourism & recreation, and water supply & quality [29]. However, this procedure has been manually done by drought experts in the NDMC, which would be inevitably affected by the subjective experience. Therefore, with employing the NLP, we expect to automate the procedure and classify the text-based information into various types of drought impacts.

1.3.2 Data and Approaches

The original data set of drought impacts based on the media were filtered and downloaded from the DIR. After cleaning up and re-organizing, the data is shown in Figure 1.5. Except for the identification code, the feature of Descrip-

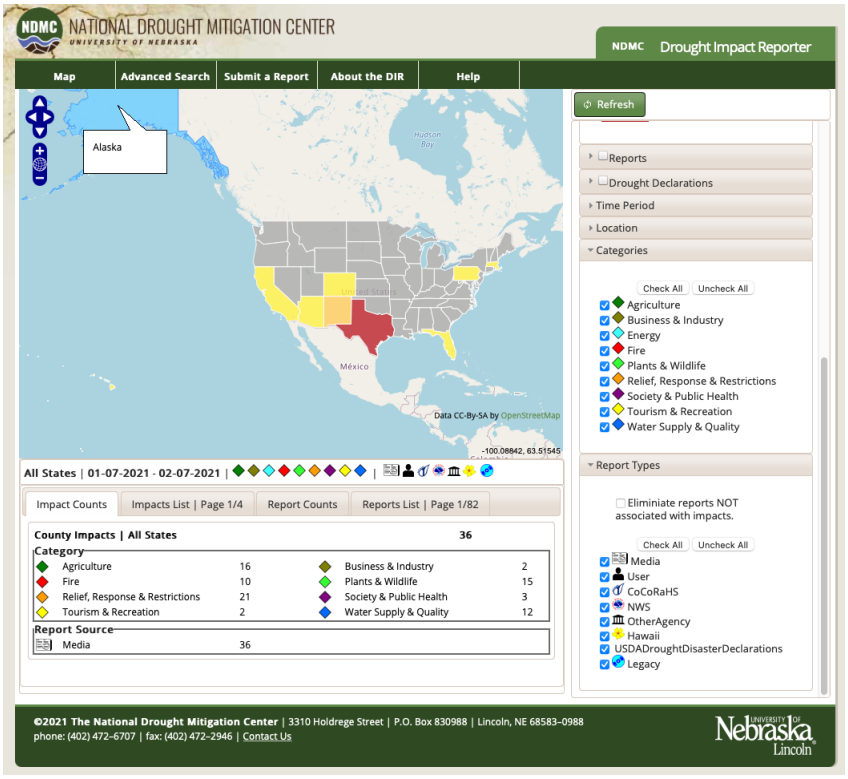

Figure 1.4: Online portal for the DIR at the NDMC website (https://droughtreporter.unl.edu/map/).

tion is the text-based drought impacts collected from media, and the column of Categories is the types of impacts. Because of the complexity of drought impacts, the corresponding types might not be unique. Therefore, compared to the first project idea, although the second one will also employ an NLP-based model, it is a multi-label classification problem, which means the same target can be assigned with more than one label [30]. The related background and approaches of the TC has been discussed above. In this project, according to the multi-label classification, we plan to employ some classical neural networks in the NLP, such as a convolution neural network (CNN) with a word embedding [31][32]. In the end, we expect to have a moderate performance on classifying the various drought impacts based on the media data.

Id	Description	Categories
1624	Voluntary water restrictions are in effect for...	Water Supply & Quality,Relief, Response & Rest...
1625	Drought hurt the lentil crop in Montana and No...	Agriculture
1626	The voluntary water conservation level remains...	Water Supply & Quality,Relief, Response & Rest...
1627	Nashville officials are struggling to find a s...	Society & Public Health,Plants & Wildlife
1628	The governor of Tennessee is seeking a secreta...	Relief, Response & Restrictions,Agriculture

Figure 1.5: Text-based drought impacts data from media on the DIR.

1.4 Conclusions

We are very interested in disaster detection in real-world life. Because this is a key issue related to the safety of people's lives and property. By applying Deep learning method into this area, we expect to seek a more efficient way to solve this task. Here, we focus our attention to the data collected from online media, which becomes the main tool people get informed. The fist idea of classifying whether the content of tweets is disaster related using NPL model, is a fundamental start of detecting disasters and affected people from social media in near-real time. After we filter the disaster related tweets from tons of posts everyday, the disaster relief team can understand the needs of affected people and the disaster impact from these selected tweets. Social media's properties of wide user base and near-real time responds helps a comprehensive and efficient collection of information, making this research very valuable in application. While the second idea of classification of drought impact based on online media report is very useful in research aspect. By using NPL machine learning method to automatically classifying raw materials, we can save the drought experts from redundant and repetitive work of manual classification. So they can have more time on more difficult and important researches.

Both ideas are TC problems. To get better accuracy, we suppose to apply different algorithm on this problem and compare different outcomes for the final model. Various preparation equipment will also be tried before training the model to find the best solution to these problems. For this part, here are some questions to TAs.

- For idea 1, since people usually use unstructured, sparse phrases and metaphors, what we can do to better recognize these words?
- For idea 2, the content of reports is pretty long. for this kind of material, what preparation equipment do you suggest to limit the irrelevant information.

Chapter 2

Milestone 2: Project Selection

2.1 Introduction

As we introduced and explained in Chapter 1, two similar topics were proposed to apply the NLP-related deep learning framework to classify text information collected from online media. We are going to choose the first project that predicts whether the tweet is related to a real natural disaster or not with the text of tweets, which is a binary text-classification problem.

The reason why we choose this project is mainly because we are inexperienced in NLP and Deep Learing (DL), we plan to gradually make progress and learn from the project. A too challenging problem beyond our capability might lead to the opposite results. The tweets-related project, as compared to the online media project, is more mature on the data set and tends to have a complete pipeline of text-based data processing and analysis. Besides, this Twitter-related project could contribute to the early warning system and the mitigation plan in managing the risk of natural disasters by providing additional information to policymakers. Moreover, the problem is the most basic binary classification that whether a tweet is related to natural disasters or not, which is good to compare the performances among different models because of the simplicity. Therefore, in the end, we choose the first proposed project: predicting whether a tweet is about natural disasters as the topic of our final project.

2.2 Problem Specification

2.2.1 Project Topic

In the final project, we will determine whether the tweets are disaster-related or not based on NLP and DL techniques, which is a binary text classification

problem. We plan to build and evaluate the performance of different feature engineering and classification models. By comparing the metrics of various models, such as the F1 score and the area under curve (AUC), we can implement a practical pipeline of developing NLP-based models and investigate models' properties at different steps and their impacts on the model performance.

2.2.2 Background and Motivation

As as widely used and nearly real-time media, monitoring Twitter on disaster information automatically has great value of real application in disaster responds. Many humanitarian agencies, like disaster relief team, are interested in monitoring Twitter. The twitter data could provide these organizations with real-time information on urgent needs of affected people, situation awareness of disaster impact, infrastructure damage et.al. In this way, they can save lives and provide aid effectively.

In thus, Twitter has been increasingly used as a source of inputting data in studying natural disasters. For instance, Sakaki et al. (2010) is a classical study that employed tweets as its only inputting data to detect the location of earthquakes in Japan based on probabilistic models [7]. Ashktorab et al. (2014) also used Twitter to extract the damages of disasters by applying various machine-learning methods, such as support vector machine, decision tree, and logistic regression [33]. While there have not been many application of NNs in this area. A recent study that Madichetty and Muthukumarasamy have done in 2020 built and compared different NNs with various word embedding methods on detecting situational information during disasters [34]. However, the tweets they inputted were in Hindi. A model based on English tweets is needed.

Therefore, in this final project, we will compare the performance of various feature engineering methods and classification models in classifying whether the English tweets are disaster-related or not. This is a challenging task, since human language is not straight-forwardly understandable by machine, not to say various expression and metaphors used by human. Some words that are usually used to describe natural disasters could be used on expressing other things, such as human feelings. For example, "blaze" can be used for a very large burning fire and an expression of anger or a euphemism of "hell". Understanding the meaning of this type of word could be challenging to simple feature engineering algorithms counting the word frequency. Sophisticated models that employ deeper neural networks and art-of-state embedding algorithms are required to identify the sentiment of the contents.

To solve this problem, we propose to use two sophisticated word embedding algorithms into deep-learning-based classification: Long Short-Term Memory (LSTM) and Bidirectional Encoder Representation from Transformers (BERT). Both methods have a degree of ability to identify the sentiment based on the context, which is expected to improve the accuracy of the classification.

2.2.3 Resources - Tools and Data sets

The employed data set from Kaggle[23] including 7503 tweets and the corresponding keyword and location. However, most entries do not contain a value for "keywords" and "location". In thus, we will only take the text of tweets as input in our model. These tweets were already labeled as 1 (disaster-related) or 0 (unrelated). Details of the data set has been shown in Figure 1.3.

We expect to develop the project based on a Python environment. Therefore, the necessary libraries for machine learning, deep learning, and NLP are required. So far, we would need the Natural Language Toolkit (NLTK), scikit-learn (sklearn), Tensorflow, Numpy, Pandas, Matplotlib, seaborn, and tokenizatin. We might need other libraries with the project progressing. There is no expense on data sets and tools because they are open-source. The primary cost of this project is the computational power on `crane`. Based on the size of data and proposed models, we expect to have at least 64 gigabyte (GB) memory, 5-10 GB available disk storage, and 512 GPU units. With more computational resources, we can train and update the models faster and more efficient.

2.3 Proposed Method 1: LSTM For Disaster Tweets

LSTM is an artificial recurrent neural network(RNN) architecture. LSTM was proposed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber, and it was gradually improved over the years. LSTM are very widely applied in Robot control, Rhythm learning, Hand writing recognition and Human action recognition et ac. In 2007, LSTM started to revolutionize speech recognition, outperforming traditional models in certain speech applications.[35] The reason for the outstanding performance of LSTM compared with traditional NN is because of its ability in persisting former information with loops in it. This is apparent if you consider how we read an essay. We understand each word based on our understanding of previous words. We don't throw everything away and only focus on a single word.

Compared with other RNN architecture, LSTM is especially capable of learning long-term dependencies, such like speech recognition, translation and sequence classification. Because it is designed to accumulate information about the past, but also decide when to forget information. To illustrate this, the architecture of a basic LSTM cell was shown in Figure 2.1. A common LSTM unit use coupled forget and input gates. In thus, LSTM only forget memories from c(t-1) when new input x(t) is added. While it only inputs new values x(t) to the state h(t) when it forgets something older c(t-1). As a result, the output is determined both by long memory (c(t-1)) and short memory h(t). The compact forms of the equations for the forward pass of an LSTM unit with a forget gate

are shown below.

$$f_t = \delta_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \delta_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \delta_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\hat{c}_t = \delta_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \hat{c}_t$$

$$h_t = o_t \otimes \delta_h(c_t)$$

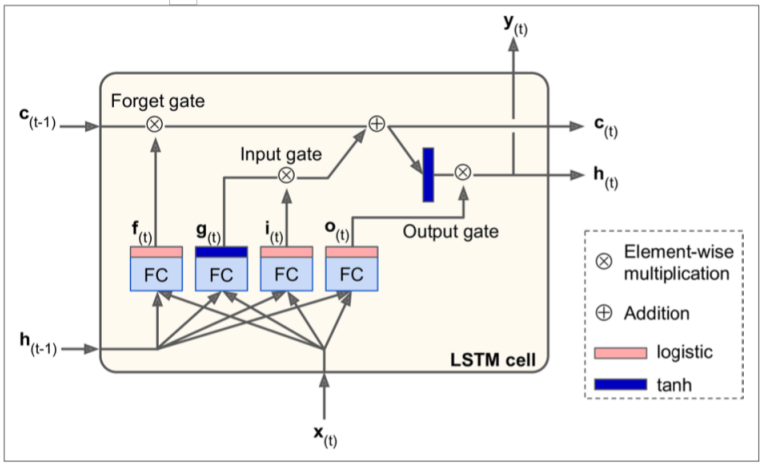

Figure 2.1: Basic Cell of LSTM

2.3.1 Data pre-processing

1. In order to get an accurate prediction, we need to remove these stop words and punctuation.

2. Also, this data is consist of tweets, which people usually post without too much concern about the grammar and letter case. They even use emojis inside the text. So, we need to normalize the letter case. We also will remove the emojis, if necessary.

3. We have different forms for one word in our daily language. For example, word "go" and "went" are talking about the same meaning, if we are not considering the tense. In this case, it would be a good idea to convert the words to its lemma form to get a better predictive result.

4. In twitter, people usually use tags, like "@" to invoke somebody's attention or "#" for a event. This is going to a factor influence the result. We are considering to remove it. Or if this sign also have something to do with the disaster, we are going to separate it as a word.

5. There also some tweets are posted in other language, like Spanish. So we also want to change those language into English.

6. Then we will do tokenization on the data set (both training data and test data).

2.3.2 Pipeline

We want to apply LSTM on this problem. Our interest laid on looking for the best model for this problem. In order to check how those models perform on this problem. Firstly, a simple Artificial Neuron Networks (ANNs) will also be tested on this data. Then we are going to construction LSTM with different architectures of various hyper-parameters..

We will map each tweet content text into a real vector domain, a popular technique when working with text called word embedding. This is a technique where words are encoded as real-valued vectors in a high dimensional space, where the similarity between words in terms of meaning translates to closeness in the vectors space. Keras provides a convenient way to convert positive integer representation of words into word embedding by an Embedding layer.

1. Preprocessing: Using the above-mentioned preprocessing method to clean data and remove noise.

2. Tokenizer: we will load the tokenizer, which give us tokens, masks and segments encode, embedding the data into vectors.

3. model building: Build the models listed below with the architecture shown.

   (a) **Simple ANNs:** This model will consists of Embedding layer, flatten layer and dense layer as output layer.

   (b) **LSTM:** In this model, Embedding layer, LSTM layer and output layer will be constructed.

4. model training: In this step, we are going to train the models mentioned in the previous step.

5. evaluation: Based on the models we trained above, we are going to test these models on the test data and find the best model.

During the experimental test, The number of neurons will be tested and Dropout layer will also be applied.

2.3.3 Evaluation

There are two common techniques for the model evaluation: holdout and cross-validation. They both work very well. Cross-validation in K-folds will be applied here. In classification problems, there are different valuation matrics, including Classification Accuracy, Confusion Matrix, Logarithmic Loss, Area under curve(AUC) and F-measures. The classification accuracy is the final goal. In the meanwhile, we will applied F1 score for the evaluation,too.

2.3.4 Timeline

Table 2.1: Timeline of proposed method 1

Period	Tasks
3-5 to 3-10 2021	Choose one method from the 2 proposed ones
3-11 to 3-31 2021	Build models: ANN and LSTM
4-1 to 4-15 2021	Tune hyper-parameters of models
4-16 to 4-20 2021	Train, validate, and test models
4-22 to 4-28 2021	Finish the reports and prepare for the final presentation

2.4 Proposed Method 2: BERT

BERT [36] is a new art-of-state model, which arouses a wide heat and interests in application of various NLP tasks, including sentiment classifications[37], Question Answering tasks[38] and Named Entity Recognition [39].BERT is a transformer[40] using attention mechanism, which is a different architecture from recurrent networks like LSTM. This transformer learn embedding by training the model to predict next-token and next-sentence probabilities. It can not only output the learnt embedding of the sentence, but also provides keywords that are important to the semantics of the sentence.

As shown in Fig2.2, The transformer is combined by two parts: Encoder (left part) and Decoder (right part). Both Encoder and Decoder are composed by stacking multiple (N_x) times of modules, which are consist mainly of Multi-Head Attention and Feed Forward layers. The scheme of Multi-Head Attention is shown in Fig.2.3, in which the left figure is a general description of the attention-mechanism that can be expressed by the following equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V \tag{2.1}$$

Here the attention can be understood as matching a query (Q, vector representation of one word in the sequence) to a key (K, vector representations of all

words in the sequence) and divided by the number of K (d_k) and mapping to a distribution of softmax. And then the distribution is scaled with a set of values (V, vector representations of all the words in the sequence). In Multi-Head Attention (Fig.2.3 right), this attention mechanism is repeated multiple times with linear projections of Q, K and V, which is done by multiplying Q, K and V by weight matrix W that are learnt during training.

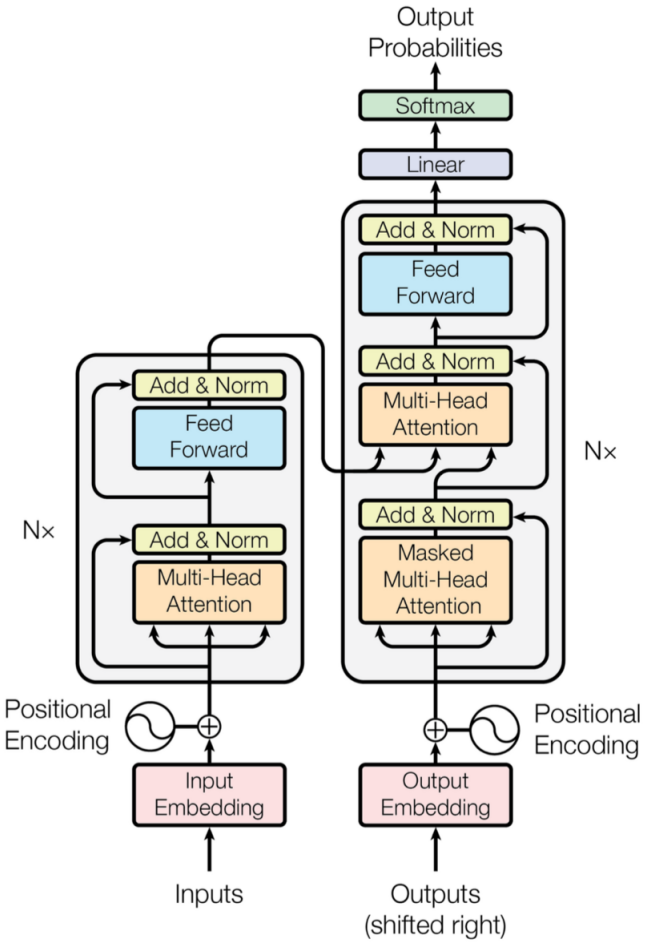

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consist attention layers running in parallel[40]

The input and output embedding are n-dimensional embedded vectors from initial Stokes. However, the decoder input will be shifted to the right by one position to avoid directly copying decoder input during training. Unlike LSTM which naturally inject the sequence of words with recurrent and covolution, the

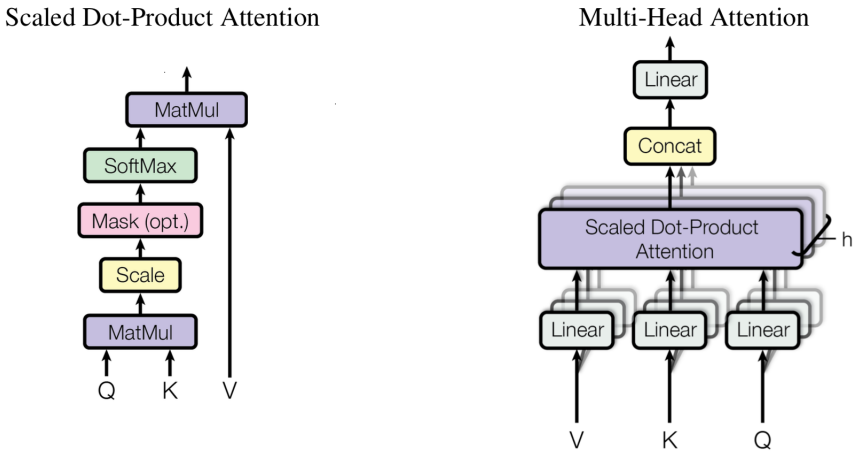

Figure 2.3: Architecture of a transformer[40]

transformer makes use of positional encoding to remember the relative position of each word/part in the sequence.

The advantage of BERT's is applying the bidirectional training of the transformer. Compared with previous methods which read a text sequence either from left to right or from right to left, it shows bidirectionally trained BERT achieves higher MNLI Dev accuracy than single-direction language models[36]. The model is pre-trained to use a "masked language model" (MLM) objective to predict next-tocken probability.

To use BERT on Classification tasks is pretty straight-forward by adding a classification layer on top of the Transformer output for the [CLS] token shown as Fig. 2.4 By fine-tuning the hyper-parameters mainly existed on the classifier's layer and keeping the values of pre-trained hyper-parameters almost same in BERT, we can train our model to predict whether the tweets are disaster-related.

2.4.1 Prepocess the data

The preprocessing methods of the data will be tried with the ones mentioned in the 1st proposed method.2.3.1

2.4.2 Pipeline of this method

As explained above, the pipeline to solve the disaster tweets classification problem is listed as below:

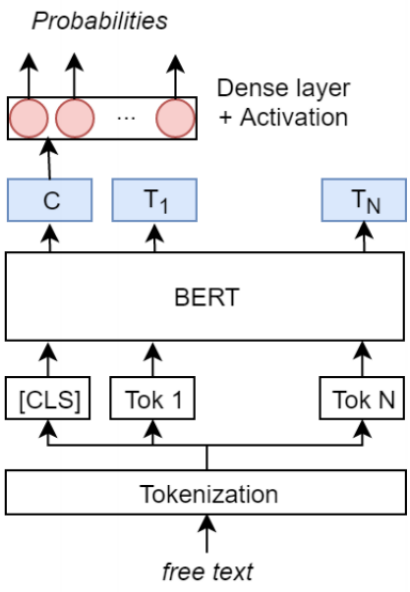

Figure 2.4: f the BERT model for multi-label classification.[41]

1. Preprocessing: Using the above-mentioned preprocessing method to clean data and remove noise.

2. Tokenizer: From the bert-layer in tensor-hub, we will load the tokenizer, which give us tokens, masks and segments encode, embedding the data into vectors.

3. Model building: Build the classification model with BERT. Various neural network classifiers can be tried as the top layer on BERT.

4. Pre-train BERT: Pre-train BERT to on Masked Langauge Modelling(MLM). This process helps the model learn natural language.

5. Fine tune model: Train BERT model to help us on the specific task. Only the output parameters are learnt from scratch and whereas the rest of the parameters are slightly fine-tuned and not that much changed which in turn makes the process faster. Then we can play with parameters of the model to achieve better accuracy.

6. Evaluation: Evaluate the classification model by calculating Classification Accuracy, Confusion Matrix, Logarithmic Loss, Area under curve(AUC) and F-measures. Classification Accuracy, Confusion Matrix, Logarithmic Loss, Area under curve(AUC) and F-measures.

2.4.3 Performance evaluation

2.3.3 The evaluation methods of the model will be same with the ones mentioned in the 1st proposed method.

2.4.4 Timeline

The timeline of this proposed method is shown as Tab. 2.2

Table 2.2: Timeline of proposed method 2

Period	Tasks
3-5 to 3-10 2021	Choose one method from the 2 proposed ones
3-11 to 3-31 2021	Build models
4-1 to 4-15 2021	Tune hyper-parameters of models
4-16 to 4-18 2021	Pre-train model
4-19 to 4-20 2021	Fine tune the model and test models
4-22 to 4-28 2021	Finish the reports and prepare for the final presentation

2.5 Conclusions

Considering the challenge of understanding tweets, both methods we proposed are to include information of surrounding text for each word. LSTM method, as an upgraded RNN, preserves long memory of the former text and short memory of the new input with loops in the cell. While BERT does not have a reccurent. It depends on attention mechanism, which encodes input sequence into a vector sequence rather than single vector. The sequence naturally include the context environment of each word.

BERT is expected to provide a better performance than LSTM because BERT allows bi-directional training. That means BERT not only includes the information of former context (left-to-right training), but also considers the relation with latter text behind the word (right-to-left training). Based on personal reading experiences, bidirectional encoding is more similar to how we understand an essay. Each word in a sentence is supported both by former and latter words. Therefore, we perfer to choose BERT for the final project.

However, the performance of these methods has to be tested in practice to confirm because it depends on dataset. If enough time is allowed, both methods will be used and compared. Various hyper-parameters will be tuned and compared to achieve the best accuracy as last.

Here, we are confused whether we have to use both proposed method in the final project. Or we are only required to choose one of them.

Chapter 3

Milestone 3: Progress Report 1

3.1 Introduction

3.1.1 Background

Natural language processing (NLP) is a field focusing on enabling the programmed computers to process and analyze large amounts of natural language data. Sentiment analysis is one challenge of NLP techniques, which means extracting emotions from raw texts. This is usually applied to automatically understand whether some users feel positive or negative on social media posts and customer reviews. In the final project, we will develop and compare two NLP models for the sentiment analysis: Long Short-term Memory (LSTM) and Bidirectional Encoder Representation from Trans-formers (BERT). The primary goal of the final project is classifying the tweeter data into related and non-related groups based on whether the contents of tweets are associated to natural disaster.

The data set employed in the project is the natural disaster tweeter data set from Kaggle, which includes more than 7,000 tweets labeled as "1" (related) or "0" (non-related) in the training set and over 3000 tweets for testing [23]. Twitter has been increasingly used as a source of inputting data in studying natural disasters. For instance, a classical study employed tweets as its only inputting data to detect earthquakes' location in Japan based on probabilistic models [7]. Another previous study also used Twitter to extract the damages of disasters by applying various machine-learning methods, such as support vector machine, decision tree, and logistic regression [33]. In contrast, there have not been many applications of NNs in natural-disaster-related NLP studies.

To identify tweets' sentiments if they indicate natural disasters, in this mile-

stone, we primarily trained and tested various LSTM-based models to compare with BERT-based models in the final report. LSTM is a method typically applied to sequence data sets, such as sentences and articles [35]. As a recurrent neural network (RNNs), LSTM takes words following sequence one by one. Thus, the output of prediction is made based on the sequence of words, enabling it to learn the dependencies among the words. Different variants of the LSTM cells and networks were developed to fit various tasks in previous studies [42]. In order to find the best model based on LSTM, we trained and compared stacked LSTM, bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU). The performance of various regularization methods, such as dropout, pooling, batch normalization, layer normalization, were also evaluated and analyzed in the report.

3.1.2 Exploratory Data Analysis

An exploratory data analysis was applied before we constructed the models to gain a better understanding of the statistics of the data. Two data sets would be employed in the project. The training set had 7613 tweets, which also included two columns: keyword and location. Keyword and location were associated with the tweets, which could be used as additional text variables in the models. The labels were given in the training data set. Figure 3.1 shows the tweets with the non-related label (0) took 57.03% of the whole training set, and tweets related to natural disasters took 42.97%. The test set removed the labels of tweets, and tweets contents, keyword, and location were included. However, there were many missing values in features of keyword and location. Figure 3.2 shows the percentages of miss values in keyword and location in two data sets. The information of location was missed about 33% in both sets, and keyword only missed about 0.8% in the training and test sets. Therefore, to reduce the effects of missing values, we dropped the feature of location and filled all missing values as 'no keyword' for the column of the keyword.

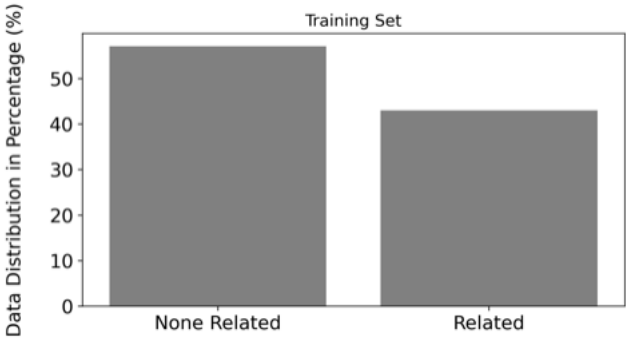

Figure 3.1: Distribution in percentage of the labels in training data set.

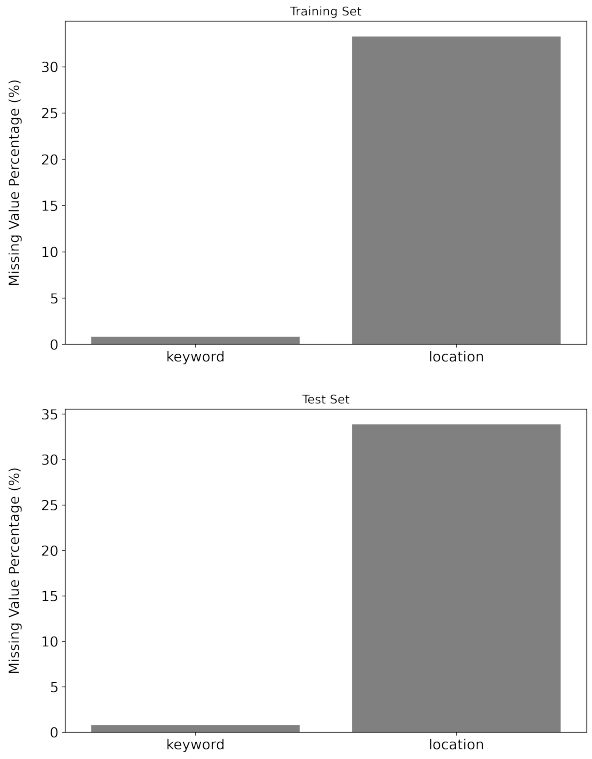

Figure 3.2: The percentage of miss values of location and keyword in training and test data set.

For the training set, two word-cloud maps were plotted to intuitively display the frequency of keywords. Figure 3.3 shows the word cloud plot for the keywords of the tweets related to natural disaster, and Figure 3.3 shows the plot of keywords with non-related labels. Overall, the keywords with different labels could potentially provide useful information to the models. For example, Figure 3.3 indicates that "forest", "fire", "wreckage", "outbreak", and "debris" were usually used in the tweets related to natural disasters. However, in tweets with non-related labels, "body", "bag", "armageddon", and "harm" were more common, which were far away from natural disasters as compared to the other group. Therefore, we would include the keywords into the model as additional information after word embedding. Although we did not expect it would significantly affect the model performance since the keywords were extracted from

the tweets, employing keyword might help emphasize the primary word in the text.

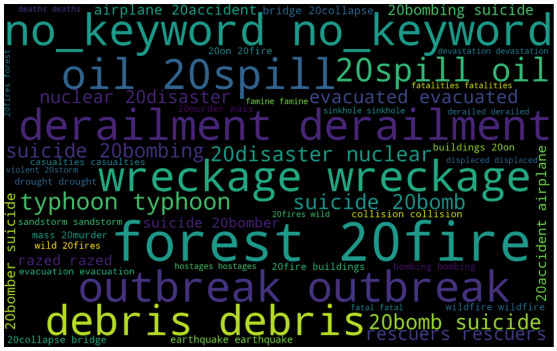

Figure 3.3: The word cloud plot of keywords in the tweets related to natural disasters.

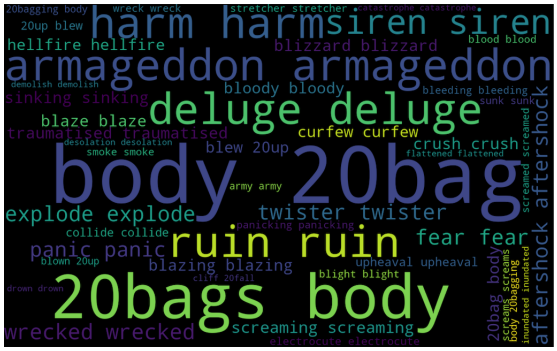

Figure 3.4: The word cloud plot of keywords in the tweets not related to natural disasters.

3.2 Experimental Setup

3.2.1 Data Description

We employed the tweets data set from Kaggle [23]. Figure 5.1 shows the structure of the data set. The details of the data and some exploratory analysis were applied and discussed in 3.1.2. In summary, the data had a training set of 7613 tweets and a test set of 3263 tweets. A feature of keyword that was extracted from the text content of tweets was also employed as an exploratory variable to the model. The data was labeled in a binary group of 0 and 1, where 0 stands for being non-related to natural disasters, and 1 for being related to natural disasters based on the content of tweets.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

Figure 3.5: The data set structure of natural disaster tweets

3.2.2 Data preparation

Training data and test data: The data set employed in this project is the natural disaster tweeter data set from Kaggle, which includes more than 7,000 tweets labeled as "1" (related) or "0" (non-related) in the training set and over 3000 tweets for testing [23].
Pre-processing: Considering the data is a tensor with type 'tf.string' in the original text representation, text data pre-processing techniques will be applied here. The applications we used are listed below:

1. **Removing HTML Tags:** Often, the unstructured text contains a lot of noise, especially if you use techniques like web or screen scraping. HTML tags are typically one of these components which don't add much value towards understanding and analyzing text. The first task of pre-processing is to remove those HTML tags.

2. **Removing Accented Characters:** Usually, in any text corpus, you might be dealing with accented characters/letters, especially if you only want to analyze the English language. Hence, we need to make sure that these characters are converted and standardized into ASCII characters. A simple example- converting é to e.

3. **Removing Special Characters:** Special characters and symbols are usually non-alphanumeric characters or even occasionally numeric characters (depending on the problem), which add to the extra noise in unstructured text. Usually, simple regular expressions (regexes) can be used to remove them.

4. **Removing Stopwords** Words that have little or no significance, especially when constructing meaningful features from the text, are known as stopwords. These are usually words that end up having the maximum frequency if you do a simple term or word frequency in a corpus. Typically, these can be articles, conjunctions, prepositions, and so on. Some examples of stopwords are a, an, the, and the like. Stopwords will be removed in this step.

As shown in Figure 5.2, after the pre-processing step, the text was more recognizable for the model to learn the sentiment from its content.

Tokenization: The next step is to tokenize the text. Here, we used python NLTK library. Once the input text was tokenized, we could create a corpus

```
The original text example:
Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all


The preprocessed text example:
deeds reason earthquake may allah forgive us
```

Figure 3.6: Text Pre-processing

and vocab in the following manner. Word cloud (Figure 5.3) and word bar (Figure 5.4) are good ways to quickly view the frequent words in the input. The distribution shows that the words, such as 'fire' and 'emergency', are very common in the data.

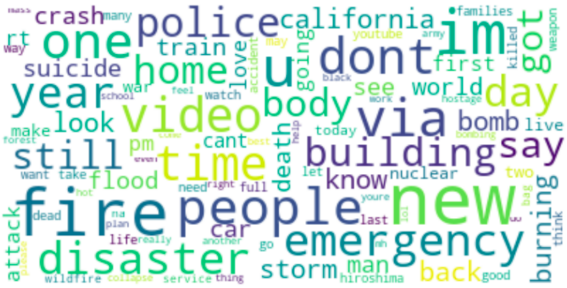

Figure 3.7: Word Cloud after tokenization

Padding: After the tokenization step, the data were padded into a numerical sequence with a certain length (64) in order to fit into the model.

3.2.3 LSTM models

In the project, we conducted experiments with various LSTM algorithms including stacked LSTM, bidirectional LSTM, and GRU. We also compared the performance of various regularization methods, such as dropout, pooling, and batch and layer normalization based on the LSTM cells. The pre-trained embedding model GloVe was employed to provide additional information because of the limited size of training data [43]. In the end, the best-performed model is shown in Figure 5.5.

1. **LSTM architectures** Based on the best model we got from the initial experiment. We changed stacked LSTM layer to GRU layer and bidirectional LSTM layer to investigate the difference. The stacked LSTM contained three stacked LSTM units that used the Tanh activation function with 100 filters. The bidirectional LSTM was an extension of the stacked LSTM. It used the same structure of LSTM units that included 3-hidden layers, Tanh activation function, and 100 filters per layer. However, for each input sequence, the bidirectional LSTM model trained two

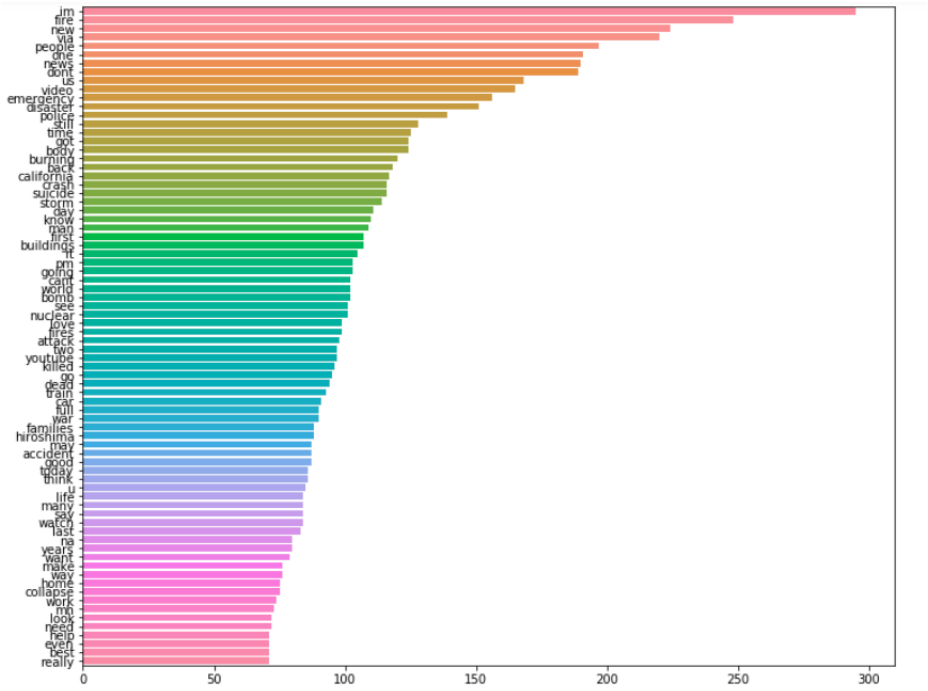

Figure 3.8: Word Bar after tokenization

Model: "model_4"

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, None)]	0	
embedding_4 (Embedding)	(None, None, 300)	6552900	input_5[0][0]
conv1d_8 (Conv1D)	(None, None, 100)	120100	embedding_4[0][0]
conv1d_9 (Conv1D)	(None, None, 100)	120100	embedding_4[0][0]
attention_4 (Attention)	(None, None, 100)	0	conv1d_8[0][0] conv1d_9[0][0]
concatenate_4 (Concatenate)	(None, None, 200)	0	conv1d_8[0][0] attention_4[0][0]
lstm_12 (LSTM)	(None, None, 100)	120400	concatenate_4[0][0]
lstm_13 (LSTM)	(None, None, 100)	80400	lstm_12[0][0]
lstm_14 (LSTM)	(None, None, 100)	80400	lstm_13[0][0]
global_max_pooling1d_4 (GlobalM	(None, 100)	0	lstm_14[0][0]
dense_8 (Dense)	(None, 128)	12928	global_max_pooling1d_4[0][0]
dropout_4 (Dropout)	(None, 128)	0	dense_8[0][0]
dense_9 (Dense)	(None, 1)	129	dropout_4[0][0]

Total params: 7,087,357
Trainable params: 534,457
Non-trainable params: 6,552,900

Figure 3.9: The best LSTM model

LSTM models with a set of normal ordered texts and inverse ordered texts respectively. Previous studies found the bidirectional LSTM would help the model learn from all available input information, but might not work well or make sense for all sequence prediction problems [44]. Therefore, we compared the LSTM and bidirectional LSTM models to test whether the bidirectional recurrent neural network would improve the model performance. Besides, we also tested another popular unit: GRU. As compared to LSTM, GRU only has two gates (reset and update gate), which makes it has a shorter memory and faster training. Because tweets are always brief due to users' habits and limitation of the characters, the GRU model might have a similar performance as compared to the LSTM model.

2. **Word embedding** Word embedding is an approach to represent the words and their relationships in the text. A word-embedding model will encode the tokens (words) into numerical vectors in the vector space. The closer the distances between two words present a similar meaning in the text. The word-embedding model can be pre-trained in advance and added to a proposed model. Therefore, in this homework, we applied a pre-trained unsupervised learning model: GloVe [43]. There are different pre-trained embedding models in GloVe. One employed model was common crawl that was trained by 840B tokens in 2.2M vocabulary in lowercase and represented in 300-dimension vector space, the other one was specifically trained by 2B tweets in 1.2M vocabulary in 200-dimension vector space [45]. In this project, we compared and analyzed the performances between two pre-trained models. Before we input the tokens into the word-embedding model, they were converted to the numerical sequence in the same length (64).

3. **Attention mechanism** A Luong-style attention layer was employed to all RNN architectures to assign weights on the work vectors and improve the models' capability to understand tokens correctly. The layer was calculated and added after the word embedding and before the LSTM, BiLSTM, and GRU layers.

4. **Regularization** Except for the dropout term and layer added onto the model, a one-dimensional global maximum pooling layer was added to the models as an optional. This layer will gather information from all LSTM and GRU units and output the maximum value. Besides, a layer normalization (LN) was added after the RNN structure. Because of the way that a RNN model trained, a classical batch normalization (BN) could not yield good results. LN, as compared to BN, computes the normalization over the features dimension. It usually has a better performance than BN in a RNN model [46]. The LN layer was add after the global pooling layer and before the dense layer. At the end, before the output dense layer, a dropout layer with the 0.3 dropout rate was also applied to overcome the overfitting.

5. **Activation functions** The default activation function (Tanh) was applied to all LSTM and GRU units to make the training process more stable and robust. For all dense layers, ReLu was used as the activation function to improve the training speed and provide sparse representations of values.

6. **Output** In the end, the sigmoid activation function was used as the output layer because the classification is binary. We used 0.5 as the threshold to transform the probabilities to labels, where the probabilities smaller than 0.5 were labeled as non-related to natural disaster (0), and the rest were related to natural disaster (1).

3.2.4 Training model

A shuffled five-fold cross-validation (CV) was employed to train the models. The mean accuracy from the five folds was output as a metric to ensure the robustness of the trained models.The number of training epoch is set as 30 for each fold of cross-validation. Early stopping is used by monitoring the validation accuracy. When the validation accuracy has no more improvement than 0.0001, the model will stop training to avoid over-fitting.

3.2.5 Performance measurement

After training the model, we calculated the binary categorical accuracy together with 95% confidential interval (CI) on test set to measure the performance since the class labels will be one-hot vectors. The CI was applied by Equation 5.1 with an assumption of the error following the Gaussian distribution, where p is the test accuracy. Accuracy is a percent value of right classifying instants in total predictions. Moreover, the confusion matrix was calculated in a classification problem, providing more comprehensive understandings of whether the model mis-classified the sentiment or not. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

$$CI = p \pm z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n}}, \alpha = 0.05 \tag{3.1}$$

3.3 Experimental Results

1. **Model architectures**: Here, we tried the three architectures and the training/validation results are show in Table 5.1. The results shows that GRU shows improvement in the training data but the validation is not as good as LSTM. The Bidirectional model is the best among these three models.

Table 3.1: A summary of LSTM-based models' performances

model	5-fold CV	Validation Accuracy
LSTM	77.1%	77.2%
GRU	77.9%	76.9%
BiLSTM	79.5%	78%

2. **Pre-trained embedding models**: Here, we tried two pre-trained word embedding models, and the results are shown in Table

Table 3.2: A summary of LSTM-based models' performances

model	5-fold CV	Validation Accuracy
LSTM with Common Crawl	77.1%	77.2%
LSTM with Tweet Crawl	79.8%	79.7%
BiLSTM with Common Crawl	79.5%	78%
BiLSTM with Tweet Crawl	80.2%	79.8%

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, None)]	0	
embedding (Embedding)	(None, None, 200)	5665800	input_1[0][0]
conv1d (Conv1D)	(None, None, 100)	80100	embedding[0][0]
conv1d_1 (Conv1D)	(None, None, 100)	80100	embedding[0][0]
attention (Attention)	(None, None, 100)	0	conv1d[0][0] conv1d_1[0][0]
concatenate (Concatenate)	(None, None, 200)	0	conv1d[0][0] attention[0][0]
bidirectional (Bidirectional)	(None, None, 200)	240800	concatenate[0][0]
bidirectional_1 (Bidirectional)	(None, None, 200)	240800	bidirectional[0][0]
bidirectional_2 (Bidirectional)	(None, None, 200)	240800	bidirectional_1[0][0]
global_max_pooling1d (GlobalMax	(None, 200)	0	bidirectional_2[0][0]
layer_normalization (LayerNorma	(None, 200)	400	global_max_pooling1d[0][0]
dense (Dense)	(None, 128)	25728	layer_normalization[0][0]
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 1)	129	dropout[0][0]

Total params: 6,574,657
Trainable params: 908,857
Non-trainable params: 5,665,800

Figure 3.10: The architecture of the best-performed BiLSTM model.

Therefore, based on the comparison above, the best model is the BiLSTM model with pre-trained GloVe embedding model on tweets. Figure 5.9 shows the architecture of the trained BiLSTM model. Figure 5.10 and 5.11 are the accuracy and loss curve on the training data set. Overall, the overfitting problem in the BiLSTM model was reduced because the application of three different regularization methods. However, the training accuracy stopped increasing at the very beginning, and the cross entropy loss didn't decrease. The accuracy on

the test data set was 79.53%, which also indicated that the model performed well on the aspect of addressing the overfitting issue. A potential reason why the model didn't conduct a efficient and validated learning process might be the insufficient model complexity. Figure 5.12 is the confusion matrix of the prediction on the test set. It reveals that the BiLSTM performed well on identifying both positive and negative classes. The recall rate is 0.74, and the precision is 0.78.

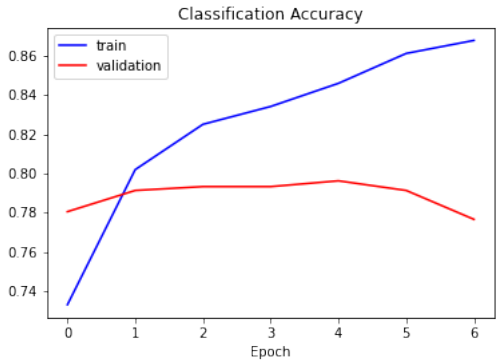

Figure 3.11: The learning curve of the best-performed BiLSTM model on training set.

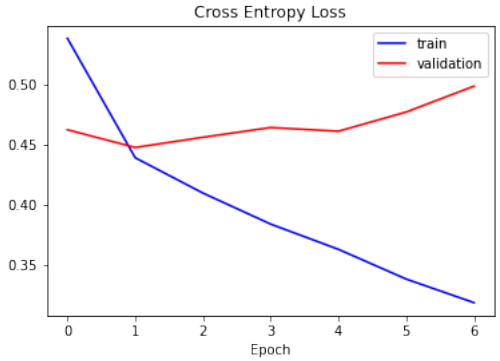

Figure 3.12: The loss curve of the best-performed BiLSTM model on training set.

3.4 Discussion

1. **GRU vs LSTM vs BiLSTM** These three models are the most common LSTM models in NLP problems. GRU is like a LSTM with a forget gate,

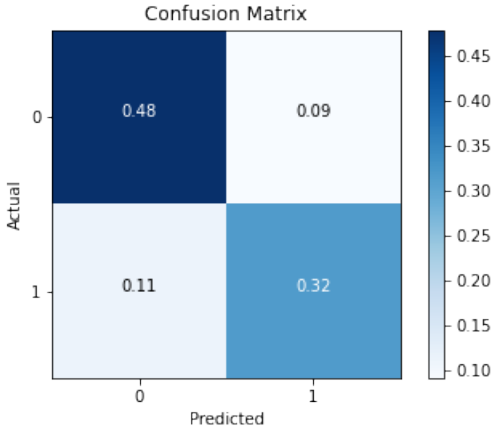

Figure 3.13: The confusion matrix of the best-performed BiLSTM model on test data.

with fewer parameters than LSTM. It shows a little bit less accurate then the LSTM and BiLSTM in this task. GRUs have been shown to exhibit better performance on certain smaller and less frequent data. However, according to the word cloud of the given tweets, some of the keywords were frequently used by users in reporting natural disasters.
BiLSTM connects two hidden layers of opposite directions to the same output. By using two time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information.[47] In our case, this method performed better than the other two.

2. **Common embedding model vs Tweets embedding model** In this scenario, we tested two different pre-trained embedding model on the LSTM and BiLSTM. According to the results shown above, we found out that the tweet based pre-trained model gave better results. It indicates that language materials would have some minor differences based on the application and sources. People who wrote those materials might have different writing habits. For example, tweets are often wrote in short sentence with many abbreviation. The pre-trained model from tweets give more accurate relations between different words.

3. **The best model's performance** Overall, the best-performed BiLSTM model had an acceptable accuracy because of the reduced overfitting issue by applying three different regularization methods. However, it is still the best result that a BiLSTM-based model could achieve based on our understanding. One of the limitations could be the relatively simple structure of the BiLSTM layers and the coarse embedding process. In the next step, the result might be improved if we could construct a deep

BiLSTM model with a finer embedding associated with the pre-trained GloVe model. Also, we noticed that the differences between LSTM and BiLSTM were insignificant. A possible reason could be the short length of tweets based on the users' habits and Twitter's restriction. Besides, bidirectional RNNs could have unstable performances on different prediction tasks of sequence data [46]. Therefore, in the next step, we will train and test the performance of BERT-based models that is another solution of the bidirectional learning.

3.5 Conclusion

In this milestone, we trained and compared three RNN models based on GRU, LSTM and BiLSTM respectively. And the BiLSTM model was better than LSTM and GRU. We also compared the GloVe pre-trained embedding models based on different data sets: common articles and Tweets. And the tweets-based pre-trained embedding model did improve the model performance. Overall, the BiLSTM model with GloVe pre-trained embedding on Tweets achieve the highest accuracy among all trained models: 79.53% on the test set. In Milestone 4, we will train and find the best BERT-based model to compared with the best BiLSTM model.

Chapter 4

Milestone 4: Progress Report 2

4.1 Introduction

Natural language processing (NLP) is a field focusing on enabling the programmed computers to process and analyze large amounts of natural language data. Sentiment analysis is one challenge of NLP techniques, which means extracting emotions from raw texts. This is usually applied to automatically understand whether some users feel positive or negative on social media posts and customer reviews. In Milestone 4, we will develop and compare Bidirectional Encoder Representation from Transformers (BERTs) with three different architectures. The primary goal is classifying the tweeter data into related and non-related groups based on whether the contents of tweets are associated to natural disaster.

The data set employed in the project is the natural disaster tweeter data set from Kaggle, which includes more than 7,000 tweets labeled as "1" (related) or "0" (non-related) in the training set and over 3000 tweets for testing [23]. Twitter has been increasingly used as a source of inputting data in studying natural disasters. For instance, a classical study employed tweets as its only inputting data to detect earthquakes' location in Japan based on probabilistic models [7]. Another previous studies also used Twitter to extract the damages of disasters by applying various machine-learning methods, such as support vector machine, decision tree, and logistic regression [33]. In contrast, there have not been many applications of NNs in natural-disaster-related NLP studies.

To identify tweets' sentiments if they indicate natural disasters, in this milestone, we primarily trained and tested various BERT-based models to compare with LSTM-based models in Milestone 3 3.1. BERT is a new art-of-state model with the best performance among other NLP models in recent research [36],

which arouses a wide heat and interests in application of various NLP tasks, such as sentiment classifications [37], Question Answering tasks [38] and Named Entity Recognition [39]. BERT is a transformer [40] using attention mechanism, which is a different architecture from recurrent networks (RNNs) like LSTM.

In this report, we applied BERT on NLP binary classification. We pursued BERT's best use for our purpose by including a regularizer, and pre-training models and changing the classifier's setting ups and hyper-parameter values. 3 architectures were examined and cross-compared. The best-performed BERT model achieved an accuracy of 83.34% on the test data set. Overall, BERT models showed slightly better results than LSTM-based models in Milestone3 3.1.

4.2 Experimental Setup

4.2.1 Data Description

We employed the tweets data set from Kaggle [23]. Figure 5.1 shows the structure of the data set. The details of the data and some exploratory analysis were applied and discussed in 3.1.2. In summary, the data had a training set of 7613 tweets and a test set of 3263 tweets. A feature of keyword that was extracted from the text content of tweets was also employed as an exploratory variable to the model. The data was labeled in a binary group of 0 and 1, where 0 stands for being non-related to natural disasters, and 1 for being related to natural disasters based on the content of tweets.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

Figure 4.1: The data set structure of natural disaster tweets

4.2.2 Data preparation

Training data and test data: The data set employed in this project is the natural disaster tweeter data set from Kaggle, which includes more than 7,000 tweets labeled as "1" (related) or "0" (non-related) in the training set and over 3000 tweets for testing [23].
Pre-processing: Considering the data is a tensor with type 'tf.string' in the original text representation, text data pre-processing techniques will be applied here. The applications we used are listed below:

1. **Removing HTML Tags:** Often, the unstructured text contains a lot of noise, especially if you use techniques like web or screen scraping. HTML

tags are typically one of these components which don't add much value towards understanding and analyzing text. The first task of pre-processing is to remove those HTML tags.

2. **Removing Accented Characters:** Usually, in any text corpus, you might be dealing with accented characters/letters, especially if you only want to analyze the English language. Hence, we need to make sure that these characters are converted and standardized into ASCII characters. A simple example- converting é to e.

3. **Removing Special Characters:** Special characters and symbols are usually non-alphanumeric characters or even occasionally numeric characters (depending on the problem), which add to the extra noise in unstructured text. Usually, simple regular expressions (regexes) can be used to remove them.

4. **Removing Stopwords** Words that have little or no significance, especially when constructing meaningful features from the text, are known as stopwords. These are usually words that end up having the maximum frequency if you do a simple term or word frequency in a corpus. Typically, these can be articles, conjunctions, prepositions, and so on. Some examples of stopwords are a, an, the, and the like. Stopwords will be removed in this step.

As shown in Figure 5.2, after the pre-processing step, the text was more recognizable for the model to learn the sentiment from its content.

```
The original text example:
Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all


The preprocessed text example:
deeds reason earthquake may allah forgive us
```

Figure 4.2: Text Pre-processing

Tokenization:The original text has to be encoded firstly with uncased tokenizer [48], which returns a dictionary with three int32 Tensors as input: input_word_ids, input_mask, and input_type_ids. Here, the input_word_ids have been randomly masked or altered by the encoder to fine tune the ML training model. Then this input was used to represent the index of each words together with its sequence in the sentence. The resulting encoder inputs have length of 128.

4.2.3 Model building

With the limited time of model training, we used a pre-trained BERT model [49] obtained from Tensorflow-hub (TF Hub) [50]. This TF Hub model uses the implementation of BERT from the TensorFlow Models repository on GitHub [51].

It uses L=4 Transformer blocks. Each block has a size of H=512, and A=8 attention heads. This model has been pre-trained for English on the Wikipedia and BooksCorpus. For training, random input masking has been applied independently to word pieces.

The Saved model provides a trainable sub-object (.mlm) with predictions for the Masked Language Model tasks that it was originally trained with. Based on that, we added a simple classifier as the output layer, which is a dense layer of a sigmoid activation function with the L2 regularization to conduct the binary classification. The encoded input was fed into the new practical model that is a sequential combination of pre-trained BERT model and classifier. The fine-tuned model, which was labeled as BERT-1, as shown by the Figure 5.6, with 28,764,161 trainable parameters.

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None,)]            0
__________________________________________________________________________________________________
keras_layer_1 (KerasLayer)      {'input_mask': (None 0           input_1[0][0]
__________________________________________________________________________________________________
keras_layer (KerasLayer)        {'encoder_outputs':  28763649    keras_layer_1[0][0]
                                                                 keras_layer_1[0][1]
                                                                 keras_layer_1[0][2]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1)            513         keras_layer[0][5]
==================================================================================================
Total params: 28,764,162
Trainable params: 28,764,161
Non-trainable params: 1
__________________________________________________________________________________________________
```

Figure 4.3: The architecture of BERT-1 model.

Based on BERT-1 model, we also proposed another 2 model with dropout. In the first architecture we applied added a dropout layer with a rate of 0.2 before the dense output layer. This model wast labeled as BERT-2. For the other architecture, labeled as BERT-3, we used a more complex classifier, which contained two dense layers. A dropout layer with a rate of 0.2 was also applied before each dense layer. The detailed architectures of BERT-2 and BERT-3 are shown in Figure 5.7 and Figure 5.8 respectively.

4.2.4 Training model

A shuffled five-fold cross-validation (CV) was employed to train the models. The mean accuracy from the five folds was output as a metric to ensure the robustness of the trained models.The number of training epoch is set as 10 for each fold of cross-validation. Early stopping is used by monitoring the validation accuracy. When the validation accuracy has no more improvement than

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None,)]                 0
_________________________________________________________________________________________________
keras_layer_1 (KerasLayer)   {'input_word_ids': (      0           input_1[0][0]
_________________________________________________________________________________________________
keras_layer (KerasLayer)     {'encoder_outputs':       28763649    keras_layer_1[0][0]
                                                                   keras_layer_1[0][1]
                                                                   keras_layer_1[0][2]
_________________________________________________________________________________________________
dropout (Dropout)            (None, 512)               0           keras_layer[0][5]
_________________________________________________________________________________________________
dense (Dense)                (None, 1)                 513         dropout[0][0]
=================================================================================================
Total params: 28,764,162
Trainable params: 28,764,161
Non-trainable params: 1
_________________________________________________________________________________________________
```

Figure 4.4: The architecture of BERT-2 model.

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None,)]                 0
_________________________________________________________________________________________________
keras_layer_1 (KerasLayer)   {'input_word_ids': (      0           input_1[0][0]
_________________________________________________________________________________________________
keras_layer (KerasLayer)     {'encoder_outputs':       28763649    keras_layer_1[0][0]
                                                                   keras_layer_1[0][1]
                                                                   keras_layer_1[0][2]
_________________________________________________________________________________________________
dropout (Dropout)            (None, 512)               0           keras_layer[0][5]
_________________________________________________________________________________________________
dense (Dense)                (None, 64)                32832       dropout[0][0]
_________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0           dense[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 1)                 65          dropout_1[0][0]
=================================================================================================
Total params: 28,796,546
Trainable params: 28,796,545
Non-trainable params: 1
_________________________________________________________________________________________________
```

Figure 4.5: The architecture of BERT-3 model.

0.0001, the model will stop training to avoid over-fitting.

4.2.5 Performance measurement

After training the model, we calculated the binary categorical accuracy together with 95% confidential interval (CI) on test set to measure the performance since the class labels will be one-hot vectors. The CI was applied by Equation 5.1 with an assumption of the error following the Gaussian distribution, where p is the test accuracy. Accuracy is a percent value of right classifying instants in total predictions. Moreover, the confusion matrix was calculated in a classification problem, providing more comprehensive understandings of whether the model mis-classified the sentiment or not. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

$$CI = p \pm z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n}}, \alpha = 0.05 \tag{4.1}$$

4.3 Experimental Results

The test and validation accuracy with 95% confidence of 3 different BERT architectures are shown in Table 5.3. All architectures reached a decent test accuracy about 80% and the 5-fold cross-validation accuracy around 85%. The architecture of BERT-2 presented a slightly better test accuracy (79.4%) than BERT-3 (78.1 %). The results show that adding more dense layers to the model like BERT-3 might not improve the performance. A potential reason might be that BERT-based models were over-fitting, which was inferred by the learning curve of BERT-2 model (Figure. 5.13), where the training accuracy was much higher than validation accuracy. The validation accuracy even tended to decrease, while the training accuracy continuously increased.

Besides, the architecture BERT-1 presented a slightly better test accuracy (81.3%) than BERT-2 (79.4%), which means the dropout layer applied in BERT-2 might not improve the performance. The reason for this result might be the twitter's text was usually short, giving little space for noise. Meanwhile, adding dropout layers may result in neglect of important information.

Table 4.1: The performances of different BERT architectures

Architecture Label	Test Accuracy	5-fold CV
BERT-1	[79.9%, 82.6%]	[81.94% , 86.66%]
BERT-2	[78.0%, 80.8%]	[92.45%, 93.31%]
BERT-3	[76.1%, 79.0%]	[85.56%, 93.14%]

Therefore, based on the comparison above, the best model is the BERT-1

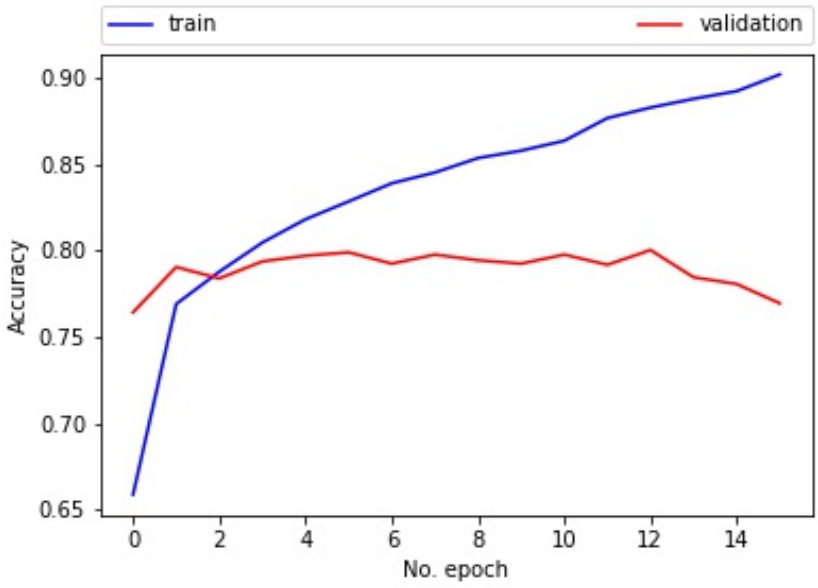

Figure 4.6: Learning curve of BERT-2 model.

model. Figure 5.14 is the confusion matrices for BERT-1, showing that BERT-1 performed well on identifying both positive and negative classes.

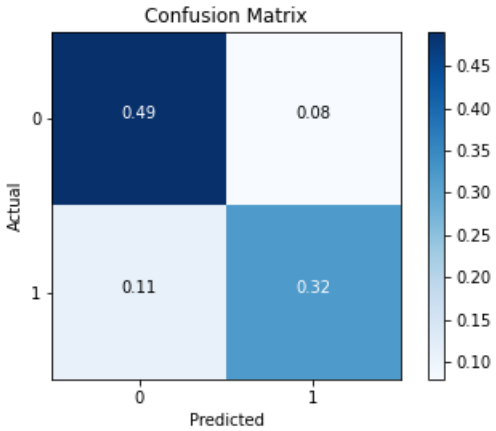

Figure 4.7: The confusion matrix of BERT-1 model.

4.4 Discussion

1. **Comparison of BERT-1, BERT-2, and BERT-3**
   The BERT models were used on classifying whether tweets are disaster related or not. All models had a significant over-fitting problem. To improve the performance, generalization methods should be applied. However, generalizing model with a dropout layer used in BERT-2 did not work as well as we expected. Moreover, since the models were over-fitting, increasing number of layers like BERT-3 was not helpful for improving the performance. A possible reason could be the short length of tweets based on the users' habits and Twitter's restriction.
   Besides, the accuracy of trials BERT-1, BERT-2 and BERT-3 were almost the same. It implies that the performance of fine-tuned BERT-based model might depend on the pre-trained model. The final performance might not be significantly affected by newly added classifier layers. Therefore, we also tried to apply other pre-trained models using Tweet database from TF Hub. However, the allowed memory resource was not enough to run it.

2. **The best model's performance**
   Overall, the best-performed BERT-1 model had an acceptable test accuracy of 81.3%, which is slightly higher than our LSTM-based models shown in Table 5.1. The best model was still over-fitted, ad increasing number of training epoch might reduce the performance. Therefore, early stopping should be applied to train the model.

4.5 Conclusion

In this milestone, to identify tweets' sentiments about the information of natural disasters, we trained and compared three BERT models by adding different architectures of classifiers layers on the pre-trained model. The performance of these three BERT models was almost the same. The best model achieved an acceptable accuracy of 81.3% on the test set. BERT-based models all had a severe over-fitting problem. However, applying generalization method to the pre-trained BERT model did not have any significant influence on the final results based on our experiments. Therefore, applying early-stopping to prevent over-fitting problem is necessary.

To further improve the performance, other pre-trained BERT models, such as the other published pre-trained or the self-trained models, should be implied and compared to the current best model. However, the allowed memory is also a limitation to us, which means the BERT model has to be small and simple to make it executable on the server. So, for our future work, we may pre-train a small BERT model based on the tweet-related dataset. We can also further improve our LSTM models (Table. 5.1) and do the hyper-parameter training again.

Chapter 5

Milestone 5: Final Report

5.1 Introduction

As a widely used and nearly real-time media, Twitter has great value of real application in disaster responds if disaster information on Twitter is monitored automatically. Many humanitarian agencies, like disaster relief team, are interested in monitoring Twitter. The twitter data could provide these organizations with real-time information on urgent needs of affected people, situation awareness of disaster impact, infrastructure damage et.al. In this way, humanitarian agencies can save lives and provide aid effectively.

In thus, Twitter has been increasingly used as a source of inputting data in studying natural disasters. For instance, a classical study employed tweets as its only inputting data to detect the location of earthquakes in Japan based on probabilistic models [7]. Twitter was also used to extract the damages of disasters by applying various machine-learning methods, such as support vector machine, decision tree, and logistic regression [33]. While there have not been many application of NNs in this area. A recent study built and compared different NNs with various word embedding methods on detecting situational information during disasters [34]. However, the tweets they inputted were in Hindi. A model based on English tweets is needed.

The first step of monitoring disaster reports on Tweet requires recognising the disaster-related information buried in the tons of tweets. This is a challenging task, since human language is not straight-forwardly understandable by machine, not to say various expression and metaphors used by human. To solve this problem, we used two sophisticated word embedding algorithms into deep-learning-based classification: Long Short-Term Memory (LSTM) and Bidirectional Encoder Representation from Transformers (BERT). Both methods have a degree of ability to identify the sentiment based on the context, which is expected to improve the accuracy of the classification. Their performance were

compared by comparing testing accuracy and confusion matrix since this is a binary classification problem.

As a result, we firstly trained and compared three LSTM models or variances: LSTM, Bidirectional LSTM (BiLSTM) and Gated Recurrent Unit (GRU), respectively. The BiLSTM model was better than LSTM and GRU. We also compared the GloVe pre-trained embedding models based on different data sets: common articles and Tweets. And the tweets-based pre-trained embedding model did improve the model performance. Overall, the BiLSTM model with GloVe pre-trained embedding on Tweets achieve the highest accuracy among all trained models: 79.53% on the test set. Then we built the classifier based on BERT pre-trained model and tried different hyper-parameters. The best-performed BERT model achieved an accuracy of 81.3% on the test data set. Therefore, BERT models showed slightly better results than LSTM-based models.

5.2 Experimental Setup

5.2.1 Data Description

We employed the tweets data set from Kaggle [23]. Figure 5.1 shows the structure of the data set. The details of the data and some exploratory analysis were applied and discussed in 3.1.2. In summary, the data had a training set of 7613 tweets and a test set of 3263 tweets. A feature of keyword that was extracted from the text content of tweets was also employed as an exploratory variable to the model. The data was labeled in a binary group of 0 and 1, where 0 stands for being non-related to natural disasters, and 1 for being related to natural disasters based on the content of tweets.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

Figure 5.1: The data set structure of natural disaster tweets

5.2.2 Data Preparation

Training data and test data: The data set employed in this project is the natural disaster tweeter data set from Kaggle, which includes more than 7,000 tweets labeled as "1" (related) or "0" (non-related) in the training set and over 3000 tweets for testing [23].
Pre-processing: Considering the data is a tensor with type 'tf.string' in the original text representation, text data pre-processing techniques will be applied here. The applications we used are listed below:

1. **Removing HTML Tags:** Often, the unstructured text contains a lot of noise, especially if you use techniques like web or screen scraping. HTML tags are typically one of these components which don't add much value towards understanding and analyzing text. The first task of pre-processing is to remove those HTML tags.

2. **Removing Accented Characters:** Usually, in any text corpus, you might be dealing with accented characters/letters, especially if you only want to analyze the English language. Hence, we need to make sure that these characters are converted and standardized into ASCII characters. A simple example- converting é to e.

3. **Removing Special Characters:** Special characters and symbols are usually non-alphanumeric characters or even occasionally numeric characters (depending on the problem), which add to the extra noise in unstructured text. Usually, simple regular expressions (regexes) can be used to remove them.

4. **Removing Stopwords** Words that have little or no significance, especially when constructing meaningful features from the text, are known as stopwords. These are usually words that end up having the maximum frequency if you do a simple term or word frequency in a corpus. Typically, these can be articles, conjunctions, prepositions, and so on. Some examples of stopwords are a, an, the, and the like. Stopwords will be removed in this step.

   As shown in Figure 5.2, after the pre-processing step, the text was more recognizable for the model to learn the sentiment from its content.

```
The original text example:
Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all


The preprocessed text example:
deeds reason earthquake may allah forgive us
```

Figure 5.2: Text Pre-processing

Tokenization: The next step is to tokenize the text. Here, since LSTM models and BERT model require different input format, we consider different ways of tokenization. For LSTM models, we used python NLTK library. Once the input text was tokenized, we could create a corpus and vocab in the following manner. Word cloud (Figure 5.3) and word bar (Figure 5.4) are good ways to quickly view the frequent words in the input. The distribution shows that the words, such as 'fire' and 'emergency', are very common in the data. After the tokenization step, the data were padded into a numerical sequence with a certain length (64) in order to fit into the model.

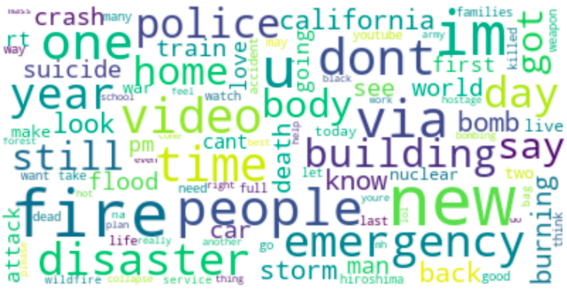

Figure 5.3: Word Cloud after tokenization

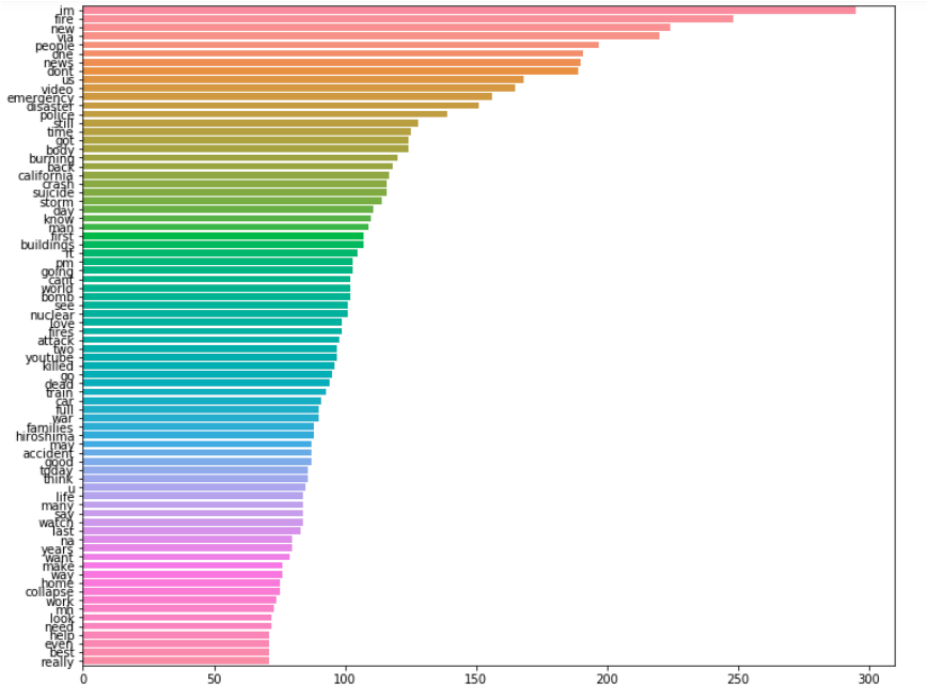

Figure 5.4: Word Bar after tokenization

As for BERT model, the original text has to be encoded firstly with uncased tokenizer [48], which returns a dictionary with three int32 Tensors as input: input_word_ids, input_mask, and input_type_ids. Here, the input_word_ids have been randomly masked or altered by the encoder to fine tune the ML training model. Then this input was used to represent the index of each words together with its sequence in the sentence. The resulting encoder inputs have length of 128.

5.2.3 LSTM Models

In the project, we conducted experiments with various LSTM algorithms including stacked LSTM, bidirectional LSTM, and GRU. We also tried to apply various regularization methods, such as dropout, pooling, and batch and layer normalization to improve the performance. The applied architecture is shown in Figure 5.5, which is mainly a stacked LSTM model containing three stacked LSTM units. Before the three LSTM units, the encoded inputs was firstly embedded and an attention layer was also included. While the output is produced by a dense layer with a sigmoid activation function since this is a binary classification problem. The detailed setting of models are explained as follows.

```
Model: "model_4"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_5 (InputLayer)         [(None, None)]            0
_________________________________________________________________________________________________
embedding_4 (Embedding)      (None, None, 300)         6552900     input_5[0][0]
_________________________________________________________________________________________________
conv1d_8 (Conv1D)            (None, None, 100)         120100      embedding_4[0][0]
_________________________________________________________________________________________________
conv1d_9 (Conv1D)            (None, None, 100)         120100      embedding_4[0][0]
_________________________________________________________________________________________________
attention_4 (Attention)      (None, None, 100)         0           conv1d_8[0][0]
                                                                   conv1d_9[0][0]
_________________________________________________________________________________________________
concatenate_4 (Concatenate)  (None, None, 200)         0           conv1d_8[0][0]
                                                                   attention_4[0][0]
_________________________________________________________________________________________________
lstm_12 (LSTM)               (None, None, 100)         120400      concatenate_4[0][0]
_________________________________________________________________________________________________
lstm_13 (LSTM)               (None, None, 100)         80400       lstm_12[0][0]
_________________________________________________________________________________________________
lstm_14 (LSTM)               (None, None, 100)         80400       lstm_13[0][0]
_________________________________________________________________________________________________
global_max_pooling1d_4 (GlobalM (None, 100)            0           lstm_14[0][0]
_________________________________________________________________________________________________
dense_8 (Dense)              (None, 128)               12928       global_max_pooling1d_4[0][0]
_________________________________________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0           dense_8[0][0]
_________________________________________________________________________________________________
dense_9 (Dense)              (None, 1)                 129         dropout_4[0][0]
=================================================================================================
Total params: 7,087,357
Trainable params: 534,457
Non-trainable params: 6,552,900
```

Figure 5.5: The architecture of LSTM-based model

1. **Word embedding** Word embedding is an approach to represent the words and their relationships in the text. A word-embedding model will encode the tokens (words) into numerical vectors in the vector space. The closer the distances between two words present a similar meaning in the text. The word-embedding model can be pre-trained in advance and added to a proposed model. Therefore, in this homework, we applied a pre-trained unsupervised learning model: GloVe [43]. There are

different pre-trained embedding models in GloVe. One employed model was common crawl that was trained by 840B tokens in 2.2M vocabulary in lowercase and represented in 300-dimension vector space, the other one was specifically trained by 2B tweets in 1.2M vocabulary in 200-dimension vector space [45]. In this project, we compared and analyzed the performances between two pre-trained models. Before we input the tokens into the word-embedding model, they were converted to the numerical sequence in the same length (64).

2. **Attention mechanism** A Luong-style attention layer was employed to all LSTM-based architectures to assign weights on the work vectors and improve the models' capability to understand tokens correctly. The layer was calculated and added after the word embedding and before the LSTM, BiLSTM, and GRU layers.

3. **LSTM units**

   Firstly, we applied LSTM units which used the Tanh activation function with 100 filters. Secondly, we used the bidirectional LSTM layer. It used the same structure of LSTM units that included 3-hidden layers, Tanh activation function, and 100 filters per layer. However, for each input sequence, the bidirectional LSTM model trained two LSTM models with a set of normal ordered texts and inverse ordered texts respectively. Previous studies found the bidirectional LSTM would help the model learn from all available input information, but might not work well or make sense for all sequence prediction problems [44]. Therefore, we compared the LSTM and bidirectional LSTM models to test whether the bidirectional recurrent neural network would improve the model performance. Thirdly, we also tested another popular unit: GRU. As compared to LSTM, GRU only has two gates (reset and update gate), which makes it has a shorter memory and faster training. Because tweets are always brief due to users' habits and limitation of the characters, the GRU model might have a similar performance as compared to the LSTM model.

4. **Regularization** Except for the dropout term and layer added onto the model, a one-dimensional global maximum pooling layer was added to the models as an optional. This layer will gather information from all LSTM and GRU units and output the maximum value. Besides, a layer normalization (LN) was added after the RNN structure. Because of the way that a RNN model trained, a classical batch normalization (BN) could not yield good results. LN, as compared to BN, computes the normalization over the features dimension. It usually has a better performance than BN in a RNN model [46]. The LN layer was add after the global pooling layer and before the dense layer. At the end, before the output dense layer, a dropout layer with the 0.3 dropout rate was also applied to overcome the overfitting.

5. **Activation functions** The default activation function (Tanh) was applied to all LSTM and GRU units to make the training process more stable and robust. For all dense layers, ReLu was used as the activation function to improve the training speed and provide sparse representations of values.

6. **Output** In the end, the sigmoid activation function was used as the output layer because the classification is binary. We used 0.5 as the threshold to transform the probabilities to labels, where the probabilities smaller than 0.5 were labeled as non-related to natural disaster (0), and the rest were related to natural disaster (1).

5.2.4 BERT Models

With the limited time of model training, we used a pre-trained BERT model [49] obtained from Tensorflow-hub (TF Hub) [50]. This TF Hub model uses the implementation of BERT from the TensorFlow Models repository on GitHub [51]. It uses L=4 Transformer blocks. Each block has a size of H=512, and A=8 attention heads. This model has been pre-trained for English on the Wikipedia and BooksCorpus. For training, random input masking has been applied independently to word pieces.

The Saved model provides a trainable sub-object (.mlm) with predictions for the Masked Language Model tasks that it was originally trained with. Based on that, we added a simple classifier as the output layer, which is a dense layer of a sigmoid activation function with the L2 regularization to conduct the binary classification. The encoded input was fed into the new practical model that is a sequential combination of pre-trained BERT model and classifier. The fine-tuned model, which was labeled as BERT-1, as shown by the Figure 5.6, with 28,764,161 trainable parameters.

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_1 (InputLayer)         [(None,)]                 0
_________________________________________________________________________________________________
keras_layer_1 (KerasLayer)   {'input_mask': (None      0         input_1[0][0]
_________________________________________________________________________________________________
keras_layer (KerasLayer)     {'encoder_outputs':       28763649  keras_layer_1[0][0]
                                                                 keras_layer_1[0][1]
                                                                 keras_layer_1[0][2]
_________________________________________________________________________________________________
dense (Dense)                (None, 1)                 513       keras_layer[0][5]
=================================================================================================
Total params: 28,764,162
Trainable params: 28,764,161
Non-trainable params: 1
_________________________________________________________________________________________________
```

Figure 5.6: The architecture of BERT-1 model.

Based on BERT-1 model, we also proposed another 2 model with dropout. In the first architecture we applied added a dropout layer with a rate of 0.2 before the dense output layer. This model wast labeled as BERT-2. For the other architecture, labeled as BERT-3, we used a more complex classifier, which contained two dense layers. A dropout layer with a rate of 0.2 was also applied before each dense layer. The detailed architectures of BERT-2 and BERT-3 are shown in Figure 5.7 and Figure 5.8 respectively.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None,)]	0	
keras_layer_1 (KerasLayer)	{'input_word_ids': (	0	input_1[0][0]
keras_layer (KerasLayer)	{'encoder_outputs':	28763649	keras_layer_1[0][0] keras_layer_1[0][1] keras_layer_1[0][2]
dropout (Dropout)	(None, 512)	0	keras_layer[0][5]
dense (Dense)	(None, 1)	513	dropout[0][0]

Total params: 28,764,162
Trainable params: 28,764,161
Non-trainable params: 1

Figure 5.7: The architecture of BERT-2 model.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None,)]	0	
keras_layer_1 (KerasLayer)	{'input_word_ids': (	0	input_1[0][0]
keras_layer (KerasLayer)	{'encoder_outputs':	28763649	keras_layer_1[0][0] keras_layer_1[0][1] keras_layer_1[0][2]
dropout (Dropout)	(None, 512)	0	keras_layer[0][5]
dense (Dense)	(None, 64)	32832	dropout[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 1)	65	dropout_1[0][0]

Total params: 28,796,546
Trainable params: 28,796,545
Non-trainable params: 1

Figure 5.8: The architecture of BERT-3 model.

5.2.5 Training Model

A shuffled five-fold cross-validation (CV) was employed to train the models. The mean accuracy from the five folds was output as a metric to ensure the robustness of the trained models.The number of training epoch is set as 30 for each fold of cross-validation. Early stopping is used by monitoring the validation accuracy. When the validation accuracy has no more improvement than 0.0001, the model will stop training to avoid over-fitting.

5.2.6 Performance Measurement

After training the model, we calculated the binary categorical accuracy together with 95% confidential interval (CI) on test set to measure the performance since the class labels will be one-hot vectors. The CI was applied by Equation 5.1 with an assumption of the error following the Gaussian distribution, where p is the test accuracy. Accuracy is a percent value of right classifying instants in total predictions. Moreover, the confusion matrix was calculated in a classification problem, providing more comprehensive understandings of whether the model mis-classified the sentiment or not. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

$$CI = p \pm z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n}}, \alpha = 0.05 \tag{5.1}$$

5.3 Experimental Results

5.3.1 LSTM Result

To improve the performance of the model, we tried different LSTM units and word embedding models.

1. **LSTM units**: Here, we tried the three kinds of LSTM units (LSTM, GRU and BiLSTM) in the architecture. Their 5-fold validation accuracy and test accuracy results are show in Table 5.1. All three models provide descent test accuracy close to 80%. The results shows that GRU has improvement in the training data but the validation is not as good as LSTM. The Bidirectional model is the best among these three models, achieving a test accuracy of 79.5%.

2. **Pre-trained embedding models**: Here, we tried two pre-trained GloVe word embedding models trained with different datasets. One model was trained with the Common Crawl dataset, while the other was trained by the Tweet Crawl dataset. Their performances are shown in Table. 5.2. No matter using which LSTM unit, LSTM or BiLSTM, the architecture with

Table 5.1: Perfomances of LSTM-based models with differient LSTM units

Model	5-fold CV	Test Accuracy
LSTM	77.1%	77.2%
GRU	77.9%	76.9%
BiLSTM	79.5%	78%

Tweet Crawl word embedding model shows better validation accuracy and test accuracy than that with Common Crawl word embedding model. This result is expected since the text data of our problem are tweets. Using a word embedding model pre-trained by the dataset on Tweets must provide the model with better ability of understanding tweet language.

Table 5.2: Performances of LSTM-based models with different word embedding models

Model	5-fold CV	Test Accuracy
LSTM with Common Crawl	77.1%	77.2%
LSTM with Tweet Crawl	79.8%	79.7%
BiLSTM with Common Crawl	79.5%	78%
BiLSTM with Tweet Crawl	80.2%	79.8%

Therefore, based on the comparison above, the best model is the BiLSTM model with pre-trained GloVe embedding model on tweets. Figure. 5.9 shows the architecture of the trained BiLSTM model.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, None)]	0	
embedding (Embedding)	(None, None, 200)	5665800	input_1[0][0]
conv1d (Conv1D)	(None, None, 100)	80100	embedding[0][0]
conv1d_1 (Conv1D)	(None, None, 100)	80100	embedding[0][0]
attention (Attention)	(None, None, 100)	0	conv1d[0][0] conv1d_1[0][0]
concatenate (Concatenate)	(None, None, 200)	0	conv1d[0][0] attention[0][0]
bidirectional (Bidirectional)	(None, None, 200)	240800	concatenate[0][0]
bidirectional_1 (Bidirectional)	(None, None, 200)	240800	bidirectional[0][0]
bidirectional_2 (Bidirectional)	(None, None, 200)	240800	bidirectional_1[0][0]
global_max_pooling1d (GlobalMax	(None, 200)	0	bidirectional_2[0][0]
layer_normalization (LayerNorma	(None, 200)	400	global_max_pooling1d[0][0]
dense (Dense)	(None, 128)	25728	layer_normalization[0][0]
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 1)	129	dropout[0][0]

Total params: 6,574,657
Trainable params: 908,857
Non-trainable params: 5,665,800

Figure 5.9: The architecture of the best-performed BiLSTM model.

Figure 5.10 and 5.11 are the accuracy and loss curve on the training data

set. The training accuracy is much higher than validation accuracy. The validation accuracy stopped increasing at the very beginning, and the corresponding cross entropy loss stopped decreasing. This means that this model has a great problem of over-fitting.

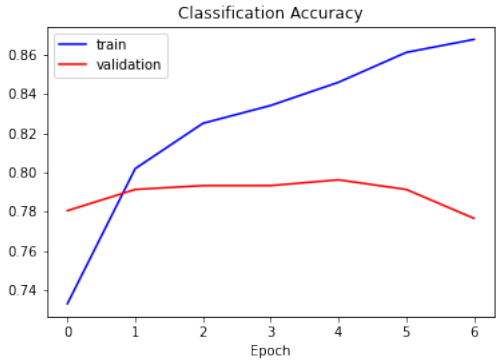

Figure 5.10: The learning curve of the best-performed BiLSTM model on training set.

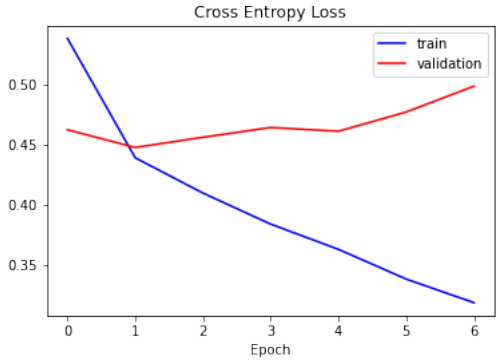

Figure 5.11: The loss curve of the best-performed BiLSTM model on training set.

Figure. 5.12 is the confusion matrix of the prediction on the test set. It reveals that the BiLSTM performed well on identifying both positive and negative classes. The recall rate is 0.74, and the precision is 0.78.

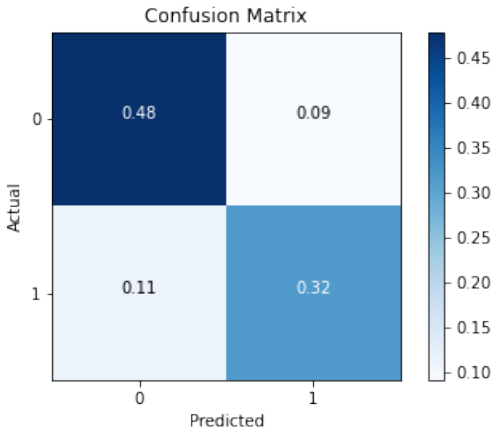

Figure 5.12: The confusion matrix of the best-performed BiLSTM model on test data.

5.3.2 BERT Result

The test and validation accuracy with 95% confidence of the 3 BERT architectures described in section 5.2.4 are shown in Table 5.3.

Table 5.3: The performances of different BERT architectures

Architecture Label	Test Accuracy	5-fold CV
BERT-1	[79.9%, 82.6%]	[81.94% , 86.66%]
BERT-2	[78.0%, 80.8%]	[92.45%, 93.31%]
BERT-3	[76.1%, 79.0%]	[85.56%, 93.14%]

All architectures reached a decent test accuracy about 80% and the 5-fold cross-validation accuracy around 85%. The architecture of BERT-2 presented a slightly better test accuracy (79.4%) than BERT-3 (78.1 %). The results show that adding more dense layers to the model like BERT-3 might not improve the performance. A potential reason might be that BERT-based models were over-fitting, which was inferred by the learning curve of BERT-2 model (Figure. 5.13), where the training accuracy was much higher than validation accuracy. The validation accuracy even tended to decrease, while the training accuracy continuously increased.

Besides, the architecture BERT-1 presented a slightly better test accuracy (81.3%) than BERT-2 (79.4%), which means the dropout layer applied in BERT-2 might not improve the performance. The reason for this result might be the twitter's text was usually short, giving little space for noise. Meanwhile, adding dropout layers may result in neglect of important information.

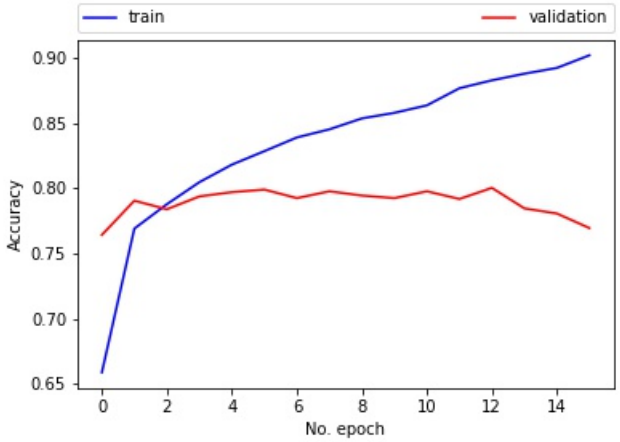

Figure 5.13: Learning curve of BERT-2 model.

Therefore, based on the comparison above, the best model is the BERT-1 model. Figure 5.14 is the confusion matrices for BERT-1, showing that BERT-1 performed well on identifying both positive and negative classes.

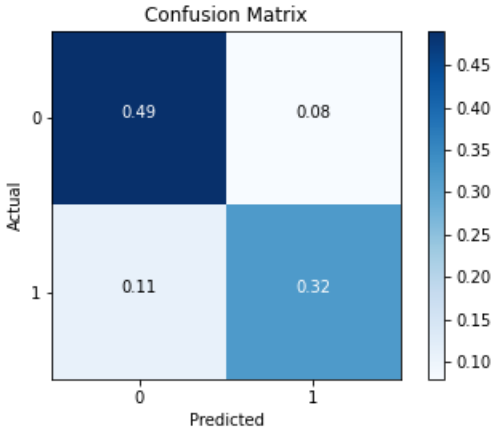

Figure 5.14: The confusion matrix of BERT-1 model.

5.4 Discussion

1. **GRU vs LSTM vs BiLSTM**

   These three models are the most common LSTM models in NLP problems. GRU is like a LSTM with a forget gate, with fewer parameters than LSTM.

It shows a little bit less accurate than LSTM and BiLSTM in this task. GRUs have been shown to exhibit better performance on certain smaller and less frequent data. However, according to the word cloud of the given tweets, some of the keywords were frequently used by users in reporting natural disasters.

Besides, BiLSTM performs better than LSTM, possibly because that BiLSTM connects two hidden layers of opposite directions to the same output. By using two time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information.[47] However, we also noticed that the differences between LSTM and BiLSTM were insignificant. A possible reason could be the short length of tweets based on the users' habits and Twitter's restriction.

2. **Common embedding model vs Tweets embedding model**

   In this scenario, we tested two different pre-trained embedding model on the LSTM and BiLSTM. According to the results shown above, we found out that the tweet based pre-trained model gave better results. It indicates that language materials would have some minor differences based on the application and sources. People who wrote those materials might have different writing habits. For example, tweets are often wrote in short sentence with many abbreviation. The pre-trained model from tweets give more accurate relations between different words.

3. **Changing the classifier layer of the BERT model**

   The 3 BERT models (BERT-1, BERT-2, BERT-3) we fine-tuned were different because of a little change of classifier layer after the pre-trained BERT model. All models had a significant over-fitting problem. To improve the performance, generalization methods should be applied. However, generalizing model with a dropout layer used in BERT-2 did not work as well as we expected. Moreover, since the models were over-fitting, increasing number of layers like BERT-3 was not helpful for improving the performance. A possible reason could be the short length of tweets based on the users' habits and Twitter's restriction.

   Besides, the accuracy of trials BERT-1, BERT-2 and BERT-3 were almost the same. It implies that the performance of fine-tuned BERT-based model might depend on the pre-trained model. The final performance might not be significantly affected by newly added classifier layers. Considering that the performance of LSTM model improves when the word embedding model pre-trained by Tweet dataset. We also tried to apply other pre-trained models using Tweet database from TF Hub. However, the allowed memory resource was not enough to run it.

4. **The best model's performance**

Among LSTM based models, the best-performed BiLSTM model had an acceptable accuracy (80.2%) because of the reduced over-fitting issue by applying three different regularization methods. One of the limitations could be the relatively simple structure of the BiLSTM layers and the coarse embedding process. The result might be improved if we could construct a deep BiLSTM model with a finer embedding associated with the pre-trained GloVe model in the future.

Overall, the best-performed model is BERT-1 model, which has an acceptable test accuracy of 81.3%, which is slightly higher than our LSTM-based models shown in Table 5.1. This model was still over-fitted, add increasing number of training epoch might reduce the performance. Therefore, early stopping should be applied to train the model.

5.5 Conclusion

In this project, we built several LSTM and BERT models to classify whether a tweet is disaster-related or not. For LSTM based models, we applied regulations, like global-max-pooling, dropout to avoid the over-fitting problem. We also included the word embedding layer and attention layer to enable the sophisticated sentiment to analyze. Besides, different LSTM units (LSTM, BiLSTM, and GRU) and pre-trained word embedding models are experimented with to pursue better performance. As a result, the BiLSTM model with a word embedding model pre-trained by Tweet Crawl data set improves the test accuracy to 80.2%. This decently performed model probably benefits from the BiLSTM's architecture, which contains two hidden layers of opposite directions connected to the same output, enabling information from both the past and future of the current time frame to decide the result. Meanwhile, the word embedding model pre-trained by dataset on Tweets makes the model stronger to understand the special language on Tweet.

Compared with LSTM models, BERT models performed slightly better. Based on a simple BERT model, we tried to improve the model's performance by changing the classifier layer. Unexpectedly, the added dropout layer and second dense layer harmed the performance. We believe that is because of the tweets' property of short length with limited information. The dropout layer can lose important information to determine the result. While a more complex classifier layer can only make the model further over-fitting and decrease the accuracy.

As a result, the best model is a BERT model, which gives a test accuracy of 81.5%. To further improve this model, we need to consider using different pre-trained BERT models. Based on the experiments of LSTM models, it is highly possible to achieve a better result if we pre-trained the BERT model with the Tweet-related data set. However, limited by the memory resource and time available, this assumption was tested in the final project.

Bibliography

[1] S. Seneviratne, N. Nicholls, D. Easterling, C. Goodess, S. Kanae, J. Kossin, Y. Luo, J. Marengo, K. McInnes, M. Rahimi, *et al.*, "Changes in climate extremes and their impacts on the natural physical environment," 2012.

[2] M. M. Council, "Natural hazard mitigation saves 2017 interim report: An independent study," *Washington, DC: National Institute of Building Sciences*, 2017.

[3] U. Ranke, "Natural disaster risk management," *Switzerland: Springer International Publishing*, 2016.

[4] S. Hallegatte, *A cost effective solution to reduce disaster losses in developing countries: hydro-meteorological services, early warning, and evacuation.* The World Bank, 2012.

[5] C. Zhang, C. Fan, W. Yao, X. Hu, and A. Mostafavi, "Social media for intelligent public information and warning in disasters: An interdisciplinary review," *International Journal of Information Management*, vol. 49, pp. 190–207, 2019.

[6] A. Ghermandi and M. Sinclair, "Passive crowdsourcing of social media in environmental research: A systematic map," *Global environmental change*, vol. 55, pp. 36–47, 2019.

[7] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*, pp. 851–860, 2010.

[8] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, p. 1–47, Mar. 2002.

[9] Y. Liu, W. Che, J. Guo, B. Qin, and T. Liu, "Exploring segment representations for neural segmentation models," *arXiv preprint arXiv:1604.05499*, 2016.

[10] J. J. Webster and C. Kit, "Tokenization as the initial phase in nlp," in *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*, 1992.

[11] G. Orphanos, D. Kalles, A. Papagelis, and D. Christodoulakis, "Decision trees and nlp: A case study in pos tagging," in *Proceedings of annual conference on artificial intelligence (ACAI)*, 1999.

[12] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "Flair: An easy-to-use framework for state-of-the-art nlp," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59, 2019.

[13] D. Ravichandran, P. Pantel, and E. Hovy, "Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 622–629, 2005.

[14] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," *Procedia Computer Science*, vol. 117, pp. 256–265, 2017.

[15] B. Xu, X. Guo, Y. Ye, and J. Cheng, "An improved random forest classifier for text categorization.," *JCP*, vol. 7, no. 12, pp. 2913–2920, 2012.

[16] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[18] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[19] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.

[20] Z.-Q. Wang, X. Sun, D.-X. Zhang, and X. Li, "An optimal svm-based text classification algorithm," in *2006 International Conference on Machine Learning and Cybernetics*, pp. 1378–1381, IEEE, 2006.

[21] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, "Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method ," *Bioinformatics*, vol. 17, pp. 1131–1142, 12 2001.

[22] J. Lever, M. Krzywinski, and N. Altman, "Classification evaluation," 2016.

[23] "Natural language processing with disaster tweets."

[24] M. J. Hayes, M. D. Svoboda, B. D. Wardlow, M. C. Anderson, and F. Kogan, "Drought monitoring: Historical and current perspectives," 2012.

[25] J. Quiggin, "Drought, climate change and food prices in australia," *Melbourne: Australian Conservation Foundation*, 2010.

[26] S. Mukherjee, A. Mishra, and K. E. Trenberth, "Climate change and drought: a perspective on drought indices," *Current Climate Change Reports*, vol. 4, no. 2, pp. 145–163, 2018.

[27] Y. Ding, M. J. Hayes, and M. Widhalm, "Measuring economic impacts of drought: a review and discussion," *Disaster Prevention and Management: An International Journal*, 2011.

[28] A. F. Van Loon, T. Gleeson, J. Clark, A. I. Van Dijk, K. Stahl, J. Hannaford, G. Di Baldassarre, A. J. Teuling, L. M. Tallaksen, R. Uijlenhoet, *et al.*, "Drought in the anthropocene," *Nature Geoscience*, vol. 9, no. 2, p. 89, 2016.

[29] D. A. Wilhite, M. D. Svoboda, and M. J. Hayes, "Understanding the complex impacts of drought: A key to enhancing drought mitigation and preparedness," *Water resources management*, vol. 21, no. 5, pp. 763–774, 2007.

[30] S. Li, "Multi label text classification with scikit-learn," Apr 2018.

[31] J. Brownlee, "Best practices for text classification with deep learning," Aug 2020.

[32] J. Brownlee, "Multi-label classification with deep learning," Aug 2020.

[33] Z. Ashktorab, C. Brown, M. Nandi, and A. Culotta, "Tweedr: Mining twitter to inform disaster response.," in *ISCRAM*, pp. 269–272, Citeseer, 2014.

[34] S. Madichetty and S. Muthukumarasamy, "Detection of situational information from twitter during disaster using deep learning models," *Sādhanā*, vol. 45, no. 1, pp. 1–13, 2020.

[35] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proceedings of the 17th International Conference on Artificial Neural Networks*, ICANN'07, (Berlin, Heidelberg), p. 220–229, Springer-Verlag, 2007.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[37] Z. Gao, A. Feng, X. Song, and X. Wu, "Target-dependent sentiment classification with bert," *IEEE Access*, vol. 7, pp. 154290–154299, 2019.

[38] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[39] F. Souza, R. Nogueira, and R. Lotufo, "Portuguese named entity recognition using bert-crf," *arXiv preprint arXiv:1909.10649*, 2019.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[41] H. Shim, S. Luca, D. Lowet, and B. Vanrumste, "Data augmentation and semi-supervised learning for deep neural networks-based text classifier," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1119–1126, 2020.

[42] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[43] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[44] J. Brownlee, "How to develop a bidirectional lstm for sequence classification in python with keras," Jan 2021.

[45] J. Pennington.

[46] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[47] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, pp. 2673 – 2681, 12 1997.

[48] TensorFlow, "bert$_e n_u ncased_p reprocess,''$ *Apr*2021.

[49] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962v2*, 2019.

[50]

[51] C. Chen, X. Du, L. Hou, J. Kim, P. Jin, J. Li, Y. Li, A. Rashwan, and H. Yu, "Tensorflow official model garden," 2020.