# Automatic classification for emergency response on social media

## (Stage Transfer Report)

Congcong Wang
Student Number: 14207199
Supervisor: Dr. David Lillis

Start date: 10th September 2018 Last DSP: 5th March 2020


University College Dublin
Dublin, Republic of Ireland
congcong.wang@ucdconnect.ie david.lillis@ucd.ie

March 13, 2020

# Abstract

The wide adoption of mobile internet-enabled devices enables people to access information in novel and interesting ways. One typical example is that people are closely connected through posting their updates on social media platforms. This has led to a dramatic increase in "user generated content", leading to social media being used for many research purposes. Recent studies indicate that the use of social media for communication and coordination during incident events (e.g. natural disasters or man-made hazards), remains to be further explored. Traditionally, the response to aid posts posted by users on social media is done through tracking and monitoring the posts manually. This type of approach becomes unrealistic and labor-consuming as the number of such posts usually explodes dramatically during a crisis.

Therefore, more and more recent efforts have directed towards the exploration of computational linguistic techniques for automatic classification of social media posts during crisis events. Although recent years have seen important progress in research for general natural language understanding, there are still two main challenges regarding its use in emergency response on social media. Firstly, the messages people post during crises tend to be short and noisy. It is difficult for computational models to understand text if it lacks context or contains noisy elements like abbreviations or misspellings. Secondly, it is important to specifically know what needs a posted message conveys so that immediate actions can be accordingly taken to answer these needs, and that responses can be prioritised towards more critical needs. This suggests that a fine-grained set of categories defining the aid needs in the posts is necessary. As a result, a single post may need to be classified into multiple categories from a large category space.

This report reviews text classification algorithms in particular for multi-label classification and also for real-time applications using social media. This review is considered to be closely related to the principal topic of this research, namely automatic classification for emergency response on social media. Having conducted this literature review, various experiments have been conducted (including participating in the Incident Streams Track from the Text REtrieval Conference), which compare and contrast the benefits of both machine learning models with hand-crafted features and representation learning based deep models for classifying crisis-related short messages into aid categories. Incorporated into these techniques, some data augmentation strategies have been applied to mitigate the imbalanced proportions of classes in a dataset and thus prevent bias. Moreover, word embeddings such as sub-word or contextualised embeddings were leveraged for better representing the messages before they are fed to the classification models.

It is expected that the future work will focus on exploring novel and advanced hierarchical multi-label classification models and pre-trained language models for learning general domain knowledge. Using a blend of state-of-the art existing and novel proposed techniques, this research ultimately aims to develop a system that is capable of adapting these to real-word situations that exhibit different characteristics. This is based on the observation that it typically cannot be said that any one set of relevant techniques will outperform all others in all circumstances.

# Table of Contents


**1 Introduction** **4**
1.1 Research Context . . . 4
1.2 Research Challenges . . . 5
1.3 Research Questions . . . 6
1.4 Report Overview . . . 6

**2 Related Work** **7**
2.1 Text Classification . . . 7
2.1.1 Machine Learning Algorithms . . . 8
2.1.2 Neural Network Algorithms . . . 10
2.1.2.1 Word Embeddings . . . 11
2.1.2.2 Deep Classification Models . . . 12
2.1.3 Text Augmentation Approaches . . . 14
2.2 Text on Social Media . . . 15
2.2.1 Text Cleaning . . . 15
2.2.2 Real Time Summarisation . . . 16
2.2.3 Crisis Detection and Tracking . . . 17

**3 Emergency Response on Social Media** **18**
3.1 Incident Streams Track (IS Track) . . . 18
3.1.1 TREC-IS Dataset . . . 19
3.1.2 Evaluation . . . 19
3.1.3 Methods Used . . . 20
3.2 More Data Collections . . . 21
3.2.1 For Unsupervised Learning . . . 21
3.2.2 For Supervised Classification . . . 22
3.2.2.1 Crisis Datasets . . . 22
3.2.2.2 General Datasets . . . 23
3.3 Summary . . . 23

**4 Progress To Date** **24**
4.1 Participation at 2019 IS Track . . . 24
4.1.1 2019 IS Edition-A . . . 25
4.1.2 2019 IS Edition-B . . . 26
4.1.3 Evaluation and Discussion . . . 27
4.2 Analysis of Word Embeddings for Classification . . . 28
4.2.1 Methodology . . . 29
4.2.2 Experimental Setup . . . 29
4.2.3 Results and Discussion . . . 31
4.3 Modules Undertaken . . . 33

**5 Future Plan** **34**
5.1 Multi-label Classification . . . 34
5.2 Development and Evaluation of Adaptive System . . . 35
5.3 Gantt Chart and Publications . . . 36

# List of Figures


1.1 Typical workflow of automatic classification for emergency response on social media. 4

2.1 A taxonomy of text classification tasks . . . 7
2.2 An illustrated example of hierarchical multi-label classification . . . 8
2.3 The general architecture of neural network models for text classification . . . 11
2.4 A taxonomy of word embeddings . . . 11
2.5 CNN as an example of sequence representation learning . . . 13
2.6 LSTM as memory cell of RNN for sequence representation learning . . . 14
2.7 Real-time Summarisation on Twitter . . . 16

3.1 The task process of TREC Incident Streams (IS) track . . . 18

4.1 System architecture at TREC IS track . . . 25
4.2 The runs at TREC-IS 2019-B among the top 13 participating runs out of 32 runs 28
4.3 The number of words (sample length) versus the number of samples (sample frequency) of the datasets - 20NewsGroup, SST-2, AAPD and Reuters. . . . 30
4.4 Embeddings (word2vec and GloVe) versus macro (20NewsGroup and SST-2) and micro (AAPD and Reuters) F1 . . . 32

5.1 Gantt chart of this research . . . 37


# List of Tables


2.1 Machine learning models comparison in text classification . . . 9

3.1 Actionable vs non-actionable information types in TREC-IS dataset. . . . 19
3.2 Major existing datasets for classification in crisis domain. . . . 22
3.3 Major existing benchmarking datasets for text classification. . . . 23

4.1 Four runs at TREC-IS 2019-A . . . 25
4.2 Generation of a critical tweet by GPT-2 . . . 26
4.3 Evaluation results of four runs at TREC-IS 2019-A . . . 27
4.4 Evaluation results of four runs at TREC-IS 2019-B . . . 28
4.5 Accuracy and macro-F1 for single-label datasets (20NewsGroup and SST-2). . . . 31
4.6 Accuracy and micro-F1 for multi-label datasets (AAPD and Reuters). . . . 32
4.7 Academic Credits Record . . . 33

# Chapter 1: **Introduction**

## 1.1 Research Context

The information age has created many ways of connecting people. Social media in particular plays an important role in today's society, resulting in an explosion of user generated content (UGC) in the last decade. Among the multiple uses of social media, its role in emergency communication and coordination has raised much interest within the research community. Emergency response agencies and humanitarian organisations continually seek more effective and rapid responses to time-critical events, such as natural disasters or human-induced hazards [27].

According to [30], an average of 50,000 people worldwide die from natural disasters annually. If critical information during an emergency is delayed, not only can it cause increased property damage but also danger to life. Hence, the timeliness of emergency response becomes a crucial factor for reducing the impact of emergency situations. The timeliness of social media enables real-time contact and communication between the people in the vicinity of the incident and the emergency aid centres. This is known as "situational awareness" (SA) [94, 95, 26], which says that effective and accurate knowledge of how an incident is unfolding helps response services to take timely preventative measures for remedying the situation.

A lot of work has been undertaken to examine SA on social media [50, 70, 74]. According to one study, 69% of people believe that emergency response operators should monitor their sites and social media accounts and respond promptly during a crisis[1]. A recent study also shows that around 10% of emergency-related posts on Twitter[2] (known as "tweets") are actionable and around 1% are critical [64]. Traditionally, the work of tracking emergencies via social media channels is done manually, where human workers filter messages to classify them as critical or not [50, 74]. Despite high precision, this is a very labor-intensive approach, particularly as the number of related messages usually increases explosively during crises. This motivates the development of computational techniques for emergency tracking and response [37, 72, 73, 105]. This process is usually done in two phases, namely initial filtering and further classification.

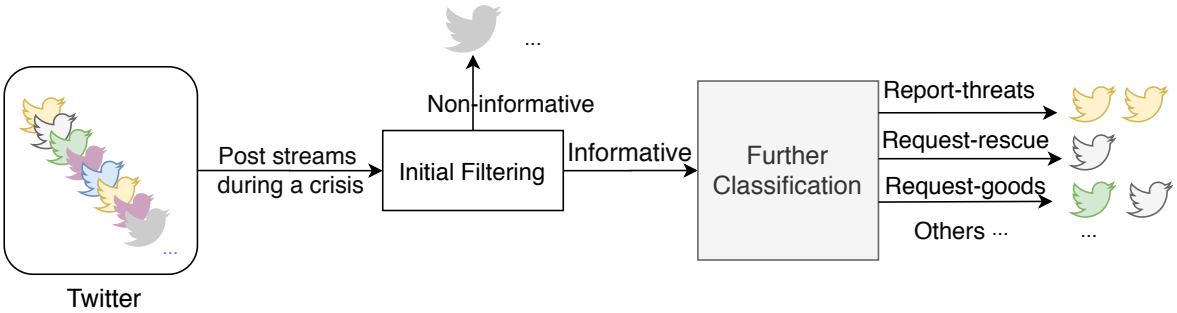



Figure 1.1: Typical workflow of automatic classification for emergency response on social media.

Taking Twitter as an example[3], Figure 1.1 presents the typical workflow of automatic classification for emergency response. It begins with a "post stream" of messages describing an ongoing crisis.

[1] https://mashable.com/2011/02/11/social-media-in-emergencies

[2] https://twitter.com

[3] Throughout this report, unless stated otherwise, Twitter is the default social media platform being discussed.

These tend to be noisy and numerous, and are initially filtered by monitoring hashtags or keywords related to the crisis. This initial filtering aims to find those that are informative to users aid needs (i.e. what assistance the users on the ground are trying to seek from emergency responders).

However, only finding the informative messages is insufficient for efficient emergency response because the categories of aid needs have different priorities (e.g. search and rescue is more critical than weather and location). This motivates the next phase: further classification is needed to identify these categories. The aid categories are generally defined in a hierarchy for best representing different aid needs. At a high level, a message can probably be about a request need, report need, etc. Within the category of request for example, it can include sub categories such as requesting search, rescue, goods, or donations. Based on the definition, each informative message after the initial filtering is further classified into one or more of the aid categories and finally pushed to corresponding operators.

## 1.2 Research Challenges

The primary focus of this work is the filtering and classification of crisis-related social media messages in real time. Based on the extensive literature review presented in Chapter 2, the main practical challenges are as follows:

- **The messages on social media tend to be short and noisy**. The messages are short because the users with the intention of seeking help on social media during crises are usually in time-critical situations. The short messages are easy for humans to understand but more difficult for machines due to the shortage of background information. For a computational model, a single message with its background information embedded as input to a model for improving its understanding of the message still remains a challenge. In addition, the messages are noisy because users usually post on social media in an informal way, particularly in the context of emergency events. The messages may contain abbreviations, out-of-vocabulary words, or emojis which harm the classification performance if not handled properly.
- **The classification needs to be fine-grained**. For emergency response as introduced, filtering for aid categories can be described as a hierarchical multi-label classification problem. First, the number of classes at low/more specific levels tends to be large because users' needs for help usually vary widely in a crisis. Given a dataset with training samples labelled by these classes, a concern raised here is the class imbalance problem. A model is easily biased to classify a message into the classes with more samples instead of those with less. Second, the hierarchical structure reflects the dependent relations between the classes at different layers. For example, a crisis-informative message falls into describing an aid of requesting ambulance. Does it indicate some information related to the aid of reporting causalities as well? To well represent the hierarchical relationships remains a challenge in literature.
- **The trade-off between latency and accuracy must be managed**. In time-critical events such as crises, the number of related messages usually increases dramatically, which makes efficiency a key requirement. Although a system developed with sophisticated learning algorithms may be capable of classifying them well in accuracy, it is still not applicable to realistic situations if the latency is too high. Given the importance of urgency in crises, the optimisation for improving the speed of the system in classification is necessary in addition to accuracy.

## 1.3 Research Questions

Given the aforementioned challenges of this research, a list of research questions (RQs) emerge:

- **RQ1: What existing and related work has been applied in the domain of crisis-related social media?** To answer this question, an extensive literature review is needed. The review helps gather datasets relevant to this research as well as identify baseline techniques used as the benchmarks to make this research measurable.
- **RQ2: How should training messages in the crisis domain be cleaned and enriched?** This will involve exploring several candidate approaches that may or may not be applicable to this particular domain. This includes learning an in-domain dictionary of out-of-vocabulary words, sub-word embeddings (including how these may handle misspellings), text augmentation to balance a biased dataset, or pre-training a model on a crisis corpus help gain a background knowledge of the message.
- **RQ3: What message representation is suitable for multi-label classification?**. This includes extracting suitable features for machine learning and exploring approaches based on deep neural networks to leverage the information of the hierarchical relationships between classes.
- **RQ4: How can the trade-off between efficiency and effectiveness be balanced?**. When it comes to real-life use cases, the choices for different techniques matter and thus an adaptive system needs to be developed. In the system, a model with high complexity performs well in accuracy while suffering from latency. How can the model be optimised for achieving better performance in latency with the minimum possible loss of accuracy? or Does ensemble approach contribute to the performance gain? When evaluating the system, what metrics are suitable for measuring the performance with regards to both its efficiency and effectiveness? or How to well quantify the performance of a model performing across crisis events (generalisation)?

## 1.4 Report Overview

The report is organised as follows. Chapter 2 presents a general review of existing techniques in relation to this research. Then, Chapter 3, which introduces this research field specifically, including the research problem domain, datasets acquired for the research and existing methods that have been applied to the research problem. Collectively, these two chapters demonstrate that RQ1 has been answered.

Chapter 4 describes two pieces of completed work that contribute towards answering RQ2 and RQ3. The first one is the participation in the TREC 2019 Incident Stream Track[4], where a number of pre-processing steps, hand-crafted features, and text augmentation strategies are explored in both machine learning models and deep models for the classification of crisis-related tweets. The second one is a comparative analysis of word embeddings in classification, which provides insights for word embedding selection in text classification tasks. Finally, Chapter 5 looks into the future and presents the research plan to answer the research questions that remain outstanding.

[4] http://dcs.gla.ac.uk/~richardm/TREC_IS/

# Chapter 2: **Related Work**

This chapter presents an extensive literature review in which baseline work is identified for this research. As the core task in this research is to classify different users' aid needs given their posted messages, this chapter first gives a review of techniques in the field of text classification (Section 2.1). In the literature, classification algorithms can be broadly divided into two categories, namely traditional machine learning algorithms (Section 2.1.1) and neural network algorithms (Section 2.1.2). The former is well known for explainability with hand-crafted features as input representations whereas the latter is known for learning deep and rich feature representations with word embeddings as input representations (Section 2.1.2.1). Since one major concern of this research is to identify one or more aid needs for a message (multi-label), one major challenge is the class imbalance problem and knowing the existing strategies for text augmentation is instructive to alleviate the problem. Hence, Section 2.1.3 reviews a number of text augmentation approaches.

After investigating text classification in depth, the review next addresses the specific field of social media. Given that this research focuses on short messages on social media, Section 2.2 examines text cleaning methods that offer insights on how to clean noisy textual data (Section 2.1.3). Section 2.2.2 surveys techniques used in real-time summarisation systems that are similar to this research in the temporal aspect. Section 2.2.3 briefly introduces the methods employed in crisis-relevant event detection and tracking, which considers the environment of the real-world deployment of an event monitoring system.

## 2.1 Text Classification

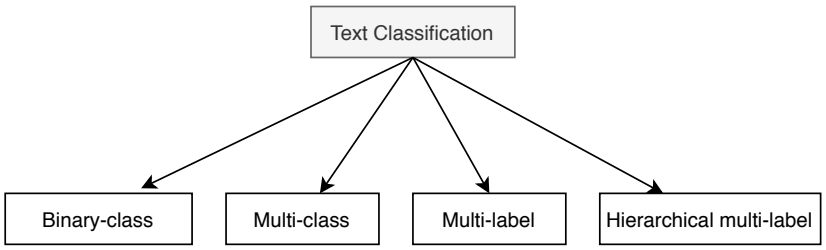



Figure 2.1: A taxonomy of text classification tasks

Text classification as an important task of natural language processing (NLP) that has been widely studied for several decades [43, 63, 111, 104]. By describing it as a supervised learning approach, the task defines input as a text sequence (sentence/document) or a training example denoted by $X :< x_1, x_2, ..., x_N >$ (where $N$ means the number of tokens in the sequence) and output as the category(s) denoted by $Y :< y_1, y_2, ..., y_K >$ (where $K$ refers to the number of elements in Y) that the sequence belongs to. Figure 2.1 presents a taxonomy of text classification. It is divided into four categories, binary-class, multi-class, multi-label, and hierarchical multi-label. To differentiate them, assume the class space is denoted by $L :< l_1, l_2, ..., l_M >$ where $M$ is the number of classes and thus $Y$ is a subset of $L$. Given $K = 1$, binary-class classification implies $M = 2$ whereas multi-class indicates $M > 2$. Unlike multi-class, multi-label classification refers to the situation when $Y$ has multiple elements in it, namely $K > 1$. Based on the definition, it is easy to deduce that multi-class multi-label classification can be described in the situation

when both $M > 2$ and $Y$ contains multiple elements, i.e. $K > 1$. Compared to multi-label, hierarchical multi-label classification not only specifies an example falling into one or more classes but a hierarchical relationship between the classes [35, 67]. Figure 2.2 illustrates an example how this type of classification is like.

The following subsections introduce machine learning algorithms and neural network algorithms that are commonly utilised for these different text classification tasks.

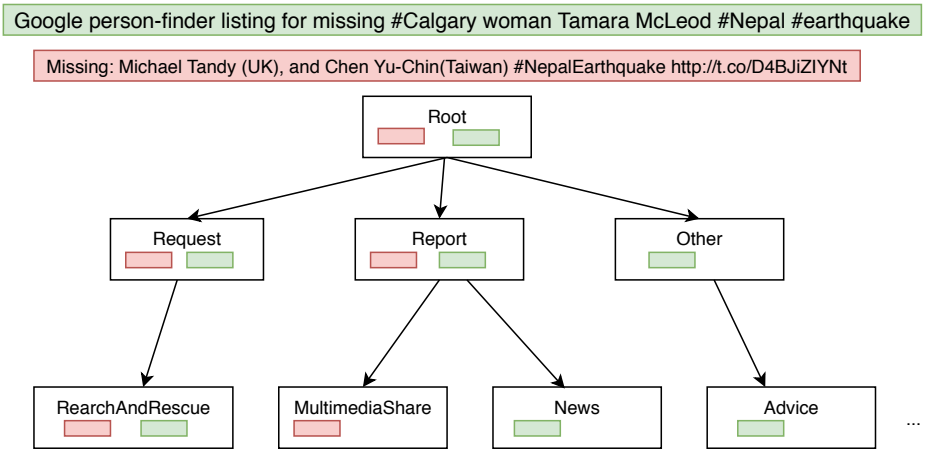



Figure 2.2: An illustrated example of hierarchical multi-label classification

### 2.1.1 Machine Learning Algorithms

Traditional machine learning (ML) algorithms are useful for text classification due to their explainable hand-crafted features and cheap computational resources. For a ML classifier, feature engineering plays an important role in representing important information for determining which class a text is likely to fall into. If the features extracted from a raw message were informative of which aid need a message is about, this would bring much performance gain in emergency response. Given the development of feature engineering in disaster tweet classification, below list a summary of them [12, 7, 69].

- **Bag-of-Words (BOW)**. Given the training set, all distinct words from the set form a vocabulary, in which each word is indexed. To represent a tweet, a vector is generated with its dimension equals the vocabulary size, in which 1 is assigned to a position if the word indexed at that position occurs in the tweet, 0 otherwise.
- **Content-based features**. These features are extracted based on the content of tweets. They could be binary values, such as the presence of URLs, hashtags or emoticons. They could also be numeric counts of special words in a tweet, such as the count of emotionally-positive or negative words, numbers, instructive words like "donate", "call", "search", etc.
- **User-based features**. These focus on the users who post the tweets, rather than the tweet contents themselves. They are considered to be important to indicate the credibility and reliability of the sources. The features could be binary (whether the users are verified on Twitter, if they are famous) or could be numeric (number of followers, friends, etc.).

Feature engineering provides representations of the training samples, and then the representations are fed to ML models for classification. Several models have been used. One of the earliest models is logistic regression (LR) [20], which is a linear classifier predicting the probabilities instead of

classes. It works by predicting the probability of a class being 0 and 1 given the sample $X$. For multi-class classification, multiple such models are trained to form multiple decision boundaries collectively determining which class $X$ is likely to be in. Although LR is easy to implement and performs well for predicting categorical outcomes [47], it suffers on non-linear problems and making predictions based on a set of independent variables. Another simple ML algorithm is Naïve Bayes (NB) which predicts the likelihood based on Bayes theorem [75]. It is widely applied for not only disaster tweet classification [69] but also general document classification [79]. It works well for text data with a very low amount of memory, and does not require parameter tuning before being trained. However, its major shortcoming is a strong assumption about the shape of the data distribution. More ML algorithms in the literature can be found in Support Vector Machines (SVMs) [61, 39] which are good for non-linear problems in comparison to LR, Decision Trees (DTs) [60] and ensemble learning algorithms such as Random Forest (RF) [11]. Table 2.1 briefly compares these algorithms with respect to their advantages and disadvantages.

| Model | Pros | Cons |
|---|---|---|
| Logistic Regression | fast, computationally cheap,<br>and no tunning,<br>no need for feature normalisation | non-linear problems,<br>make predictions based on a<br>set of independent variables |
| Naïve Bayes | easy, fast, computationally cheap,<br>and no tunning,<br>works very well with text data | a strong assumption about the<br>shape of the data distribution |
| Support Vector Machine | non-linear problems<br>robust against overfitting | computationally expensive,<br>kernel choice difficulty,<br>high dimension curse |
| Decision Tree | fast for training and prediction<br>works well for categorical features | out-of-sample prediction issue,<br>easily overfitting,<br>sensitive to noise in data |
| Random Forest | fast to train<br>ensemble learning<br>reduced variance | slow to predict if many trees,<br>easily overfitting,<br>choose no. of trees at forest |

Table 2.1: Machine learning models comparison in text classification

Most of these ML methods can be applied to a text classification task with low requirements for both memory and time[1]. However, these methods are originally designed for single label classification problems. As they evolved, many solutions to adapt them to multi-label classification have been developed. The following lists some approaches for multi-label classification.

- **Binary Relevance** [107]. This method decomposes a multi-label classification problem into multiple independent classification problems. Namely, each class is assigned a classifier that predicts whether an instance is a member of the class. Eventually, the instance is assigned a set of labels based on the predictions of all individual classifiers, and so each label is treated independently.
- **Label Powerset** [15]. In contrast to Binary Relevance, this method takes into account the label correlations by mapping each combination of labels into a single label, where a single label classifier is enough to make predictions. This leads to an imbalanced dataset and high computational complexity if there are too many classes associated with few examples.
- **Classifier Chain** [81]. Like Binary Relevance, this method learns multiple classifiers for each class. However, it links all the classifiers in order to make predictions for an instance. For example, its membership of label $A_1$ predicted by classifier $B_1$ is used as a feature for classifier $B_2$ predicting its membership of label $A_2$, and so on. The accuracy of this

[1]Implementations are available in frameworks such as Python's *scikit-learn* (`https://scikit-learn.org/`).

approach heavily depends on the order, leading to the difficulty of selecting the order in which individual classifiers are linked.

To summarise, for different types of text classification, ML algorithms are still applicable due to their interpretability, despite the superior performance often achieved by neural network algorithms (deep learning). One major difference between machine learning and deep learning lies in feature engineering and representation learning. For deep learning, there is no need for crafting features as in ML algorithms but raw data is fed to a neural network algorithm to learn feature representations automatically. A review of neural network based algorithms for text classification follows.

### 2.1.2 Neural Network Algorithms

Deep learning has become known for its powerful performance in learning multiple layers of feature representations of the data [52]. It learns feature representations through an artificial neural network consisting of layers of "neurons", which are are functions transforming an input to output. Taking text as an example, the network usually accepts the representation of a sequence (e.g. numeric vectors such as word embeddings) as the input in the first layer. Then it passes the representation forward to generate a new representation in the next layer by applying weights to the old representation (feed-forward). The weights between layers are learned based on a defined objective function, which calculates the distance between the predicted label and the actual label, such as cross entropy [40] or mean square error [13]. The objective aims to optimise the weights so that the distance is minimised. The optimisation process is done by a process of gradient descent which takes derivatives of the objective function with respect to the weights for changing the weights (back-propagation). Common optimisation algorithms include stochastic gradient descent (SGD) [82], Adam [45] and Adagrad [24]. For more details of gradient descent optimisation algorithms, see [82].

To present a more specific example, Figure 2.3 demonstrates the general architecture of neural network models for text classification. First, the text indexer indexes the raw sequence $X$ before it goes to the embedding layer. Next, the tokens $X$ are represented by vectors through the embedding layer. After the tokens are embedded, a matrix is fed to the next module which here is called the sequence encoder. Given the classification objective, the sequence encoder aims to learn a good representation for the sequence: $S$. After being encoded, $S$ is forwarded to a fully connected layer denoted by $g$ to output the logits cross all labels. This is then converted to a probability distribution to capture the probability of sample $X$ belonging to label $l_i$. For single-label classification, this is estimated by the softmax function, whereas sigmoid is used for multi-label classification (with a label being predicted for the sample if the estimated probability is larger than 0.5).

Regarding the optimisation of the back propagation process, the objective function calculates the distance between the estimated probability distribution and ground truth. In binary classification, where the number of classes $M$ equals 2, the distance can be defined by a categorical cross-entropy loss function ($BinaryCELoss$). For multiclass classification (i.e. $M > 2$), it can be defined by a separate loss for each class label, with the result being summed ($MultiCELoss$). For multi-label classification, the objective can be defined by a binary cross entropy-loss function ($BCELoss$).

When using neural networks for text classification, two unanswered questions are how the word representation is learned and how sequence representation is learned through neural networks. The following two sections reveal important techniques for learning word embeddings and major deep models for sequence learning.

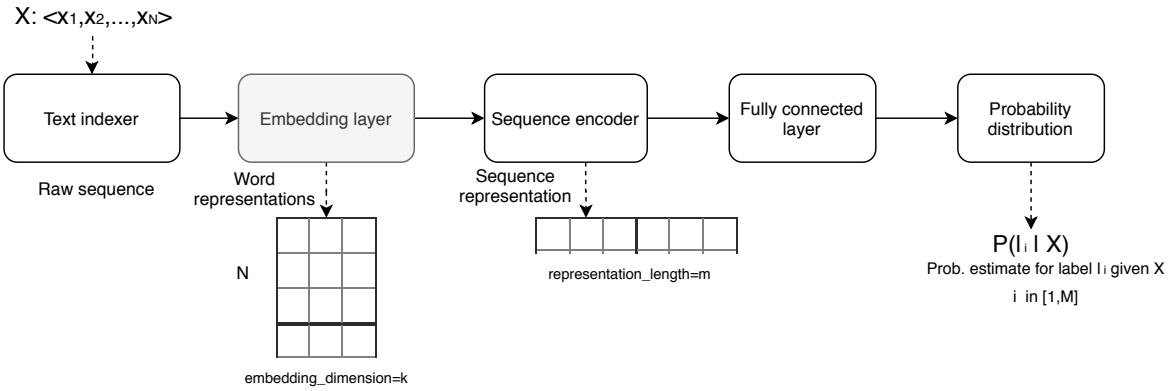



Figure 2.3: The general architecture of neural network models for text classification

#### 2.1.2.1 Word Embeddings

Word embedding is a way to represent a word with fixed-length vectors of continuous real numbers [65, 66, 9, 76]. It maps a word in a vocabulary to a latent vector space where words with similar contexts are in proximity. Through word embedding, a word is converted to a vector that summarises the word's both syntactic and semantic information. Word embeddings are often used as input feature representations for neural network models in text classification, which corresponds to the embedding layer in Figure 2.3. Figure 2.4 presents a taxonomy of word embeddings. Broadly speaking, there are two main types of word embeddings, namely context-independent and context-dependent embeddings. The typical examples of the former are word2vec [66], GloVe [76] and FasText [10]. These are known as classic word embeddings, which learn representations through language model (LM) based shallow neural networks or co-occurrence matrix factorisation [106]. The learned representations are characterised by being distinct for each word without considering the word's context. Hence, there are usually pre-trained and stored in the form of downloadable files which can be directly applied to text classification tasks.

In comparison to context-independent word embeddings, context-dependent methods learn different embeddings for the same word with different contextual use. For example, for the polysemy "bank", its embedding changes depending on whether it is used in a river-related context or finance-related one. This feature has made this type of embeddings to become the mainstream. Below gives a short introduction to some specific word embedding approaches.

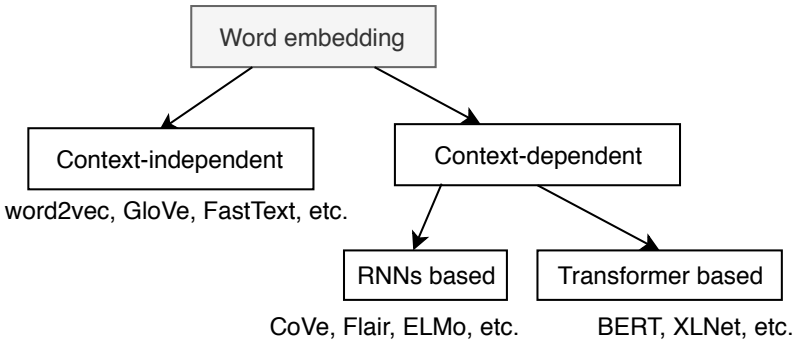



Figure 2.4: A taxonomy of word embeddings

**Classic embeddings**: This type of embeddings are pre-trained over very large corpora and shown to capture latent syntactic and semantic features.

- **word2vec** [66]. This applies either of two model architectures to produce word vectors, namely continuous bag-of-words (CBOW) and skip-gram (SG). Both methods are trained

based on a neural prediction-based model. CBOW trains a model that aims to predict a word given its context, while SG does the inverse, namely to predict the context given its central word.

- **GloVe** [76] learns efficient word representations by performing training on aggregated global word-word co-occurrence statistics from a corpus. It is known for learning good general language features by capturing words' co-occurrences globally across corpora.
- **FastText** [10] learns word representations through a neural LM. Unlike GloVe, it embeds words by treating each word as being composed of character n-grams instead of a word whole. This allows it to not only learn rare words but also out-of-vocabulary words.

**Contextualised embeddings**: Compared to classic embeddings, this type of embeddings are known for capturing word semantics in context.

- **ELMo** [77] learns contextualised word representations based on a neural LM with a character-based encoding layer and two biLSTM layers (these are discussed later in Section 2.1.2.2). The character-based layer encodes a sequence of characters of a word into the word's representation for the subsequent two biLSTM layers that leverage hidden states to generate the word's final embedding.

  **Flair** [3]. This trains a model for producing contextualised word embeddings using neural character LM (1-layer biLSTM), leading to lower computation resources and stronger character-level features. Although its effectiveness has been recognised in sequence labelling tasks, its performance in text classification remains unexplored.

  **BERT** [21] is a recently proposed Transformer-based [93] language representation model trained on a large cross-domain corpus. Unlike ELMO that learns representations through bidirectional LMs (i.e., simply the combination of left-to-right and right-to-left representations), BERT applies a Masked LM to predict words that are randomly masked. Instead of the recurrent learning of a sequence, BERT uses the Multi-Head Self-Head Attention mechanism [93, 6] to learn global dependencies between the words of a sequence, leading to being more parallelizable and superior performance. Through a process of fine-tuning, BERT has been demonstrated to achieve state-of-the-art results for a range of NLP tasks. Other similar embeddings optimised for BERT include XLNet [103], DistilBERT [84], RoBERTa [57] and ALBERT [46].

#### 2.1.2.2 Deep Classification Models

In a neural network, the layers can be customised and connected in different ways to learn representations of data for a target problem. This brings many variations of deep models. This section introduces two commonly-practised models for learning sequence representations, namely, convolutional neural networks (CNN) and long short-term memory based recurrent neural network (LSTM-RNN).

**CNN** [106, 110] is a variant of feed-forward neural network originally employed in the field of computer vision. It consists of multiple convolutional layers, each of which acts as a local filter to extract features from the input by sliding (or convolving) it, like the cells in visual cortex of human brain receiving local features (e.g. light or contour, known as receptive fields) of an input image. Figure 2.5 shows the workflow of CNN as an example of sequence representation learning. In the first stage, the input matrix (i.e. the word representations), is converted to feature/activation maps through the convolution layer. The convolution process consists of element-wise multiplications between the input matrix and the defined local filters which are analogous to windows. Mathematically, the local filters are simply arrays of weights, and its number and region size can be customised. In this example, there are two sizes of such filters - $2$ and $3$, each of which has

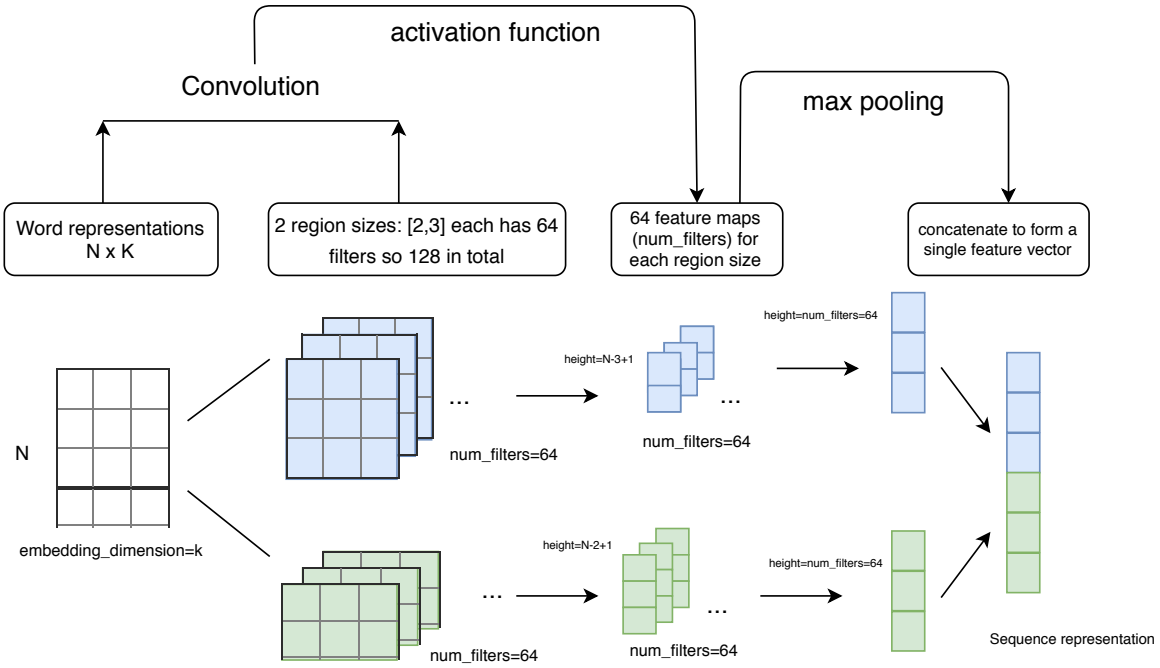



Figure 2.5: CNN as an example of sequence representation learning

$64$ filters. For the $64$ filters with region size $3$, each slides from the top to bottom of the input matrix (element-wise and sum) and thus output 64 feature maps (or representation) consisting of vectors with each size being $N - 3 + 1$. This is the same for the $64$ filters with region size $2$, outputting $64$ feature maps consisting of vectors with each size being $N - 2 + 1$. Following the convolutional layer, a pooling (or sub-sampling) layer is applied to reduce the spatial size of the representation, while retaining the most salient information of the previous representation. For instance, max pooling in Figure 2.5 refers to the sub-sampling operation that takes only the maximum number out of the vectors of 64 filters of each size and thus outputs a single vector with height being $64$ (i.e. the number of filters). In practice, the convolution and pooling processes are usually conducted multiple times to learn deeper representations. In this simple example, the sequence representation is eventually generated by concatenating the two vectors from the pooling layer. CNN is a widely-used model as a feature extractor for learning representations in text classification [19, 23, 44, 108] because of its focus on finding local clues of a textual data. Taking a document as an example, there are often phrases or n-grams that are in different places but are very informative of which topic the document is in. CNN is good at finding such local indicators, irrespective of the positions.

**LSTM** [32, 112] is a memory-based unit in a RNN. Although there are many other variations of RNN-based memory units, such as GRU [16], only LSTM is detailed below due to its popularity and wide use. Figure 2.6 shows the uni-directional (left-to-right) workflow of LSTM as memory cell of RNN for sequence representation learning. In a general sense, it encodes the input matrix to a sequence representation. At each time step $t$ in a token vector $v_t$, it summarises important information in the sequence so far as a hidden state $h_t$, using the previous hidden state $h_{t-1}$ as input. It does so by remembering or forgetting information through a set of transition functions (or so-called gates) in the unit[2]. Ultimately it summarises the sequence by outputting a single vector at the last time step.

Unlike a LSTM, which summarises information in one direction from left to right, bidirectional LSTM (biLSTM) sees the context of a token at time step $t$ in both left-to-right and right-to-left directions. In biLSTM, the right-to-left process applies the same the transition functions as

[2]For the mathematical equations, see `https://pytorch.org/docs/stable/nn.html`

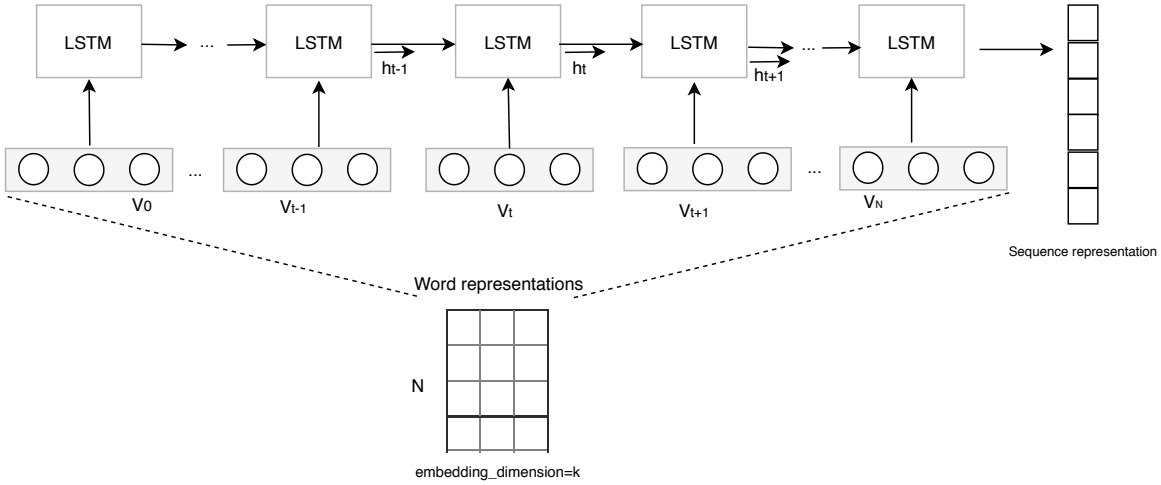



Figure 2.6: LSTM as memory cell of RNN for sequence representation learning

in the left-to-right process. In practice, there could be multiple layers of LSTM units stacked upon each other to learn deeper representations of the input data. In addition, depending on the problem domain, there exists different ways of generating final sequence representation, such as concatenation of the mean of the outputs at each time step (bag-of-means) [41] or only the output at the last time step as in the case of Figure 2.6.

Given the extensive studies in recent years on deep models for text classification, CNN and LSTM are just two examples of many in the literature. Due to the limited space of this review, these are not described in depth. For example, [111] combines CNN with LSTM for text classification; [104] applies hierarchical attention networks; [112] improves text classification by integrating bidirectional LSTM with two-dimensional max pooling; [77] applies a bi-attentive classification network (BCN) with pre-trained contextualised ELMo embedding, achieving state-of-the-art performance in text classification. More recently, transfer learning has gained great success in language understanding, which pre-trains on large textual corpora to gain a pool of general knowledge through unsupervised learning and then fine-tunes on the given training dataset (knowledge transfer) through supervised learning for a specific language task such as text classification. The major efforts into transfer learning include ULMFiT [33], ELMo [77], BERT[21], ALBERT[46].

### 2.1.3 Text Augmentation Approaches

One main challenge of automatic classification for emergency response is the class imbalance problem. Given the classes (i.e. types of aid need) assigned to training samples, the majority classes often fall into the messages with less urgent aid needs, while the messages with high urgency often belong to the minority classes [64]. Through supervised learning, it is important for a model to learn more of the highly critical messages so that better predictions can be made for finding actionable messages among unseen upcoming crisis-related streams. The class imbalance concern has sparked much research in exploring text augmentation (TA) techniques. Broadly speaking, the TA techniques in the literature mainly fall into two categories: over-sampling and NLP techniques.

**Over-sampling Techniques.** Over-sampling augments the minority class by increasing its frequency in dataset in the way of sampling. Random over-sampling randomly samples an instance from the minority class and replicates it, and this is repeated until the number of minority instances matches that of the majority class. This method is easy to understand, efficient to implement, and advantageous for no information loss. However, it possibly leads to over-fitting. Another

evolved method is known as cluster-based over-sampling [85], which applies the K-means clustering algorithm to identify clusters in both the majority and minority classes. Each cluster is then oversampled so that all clusters of the same class reach an equal number of instances. Compared to random over-sampling, this method does not guarantee a total equal number of instances of each class and thus likelihood of over-fitting. However, it is still an issue when over-fitting occurs. To avoid the over-fitting problem, synthetic minority over-sampling technique (SMOTE) was developed for imbalanced data [14]. SMOTE works first by distinguishing the minority class instances from the majority ones and then takes a subset of the minority ones for distance computation. The distance is computed by the space length of a line between any a combination of two data points in the subset and finally synthetic instances are generated randomly somewhere in the space lines. Compared to random over-sampling, SMOTE does not introduce loss of useful information as well as mitigating the problem of over-fitting. Despite the advantages, SMOTE is likely to bring noise to the original dataset due to its lack of consideration for neighbouring examples from other classes. Modified SMOTE (MSMOTE) [34] is a variant of SMOTE that further explores optimisation strategies to the drawback of SMOTE.

**NLP Techniques.** Data augmentation techniques in NLP have been developed for generating more textual samples to boost model performance. For the class imbalance problem, these techniques can be applied to generate synthetic instances for the minority class from a linguistic perspective. The central idea of this kind of techniques is that an instance for the minority class is generated based on all the existing instances in the minority class and ideally the generated instance has to keep the same semantic meaning as the existing instances. Otherwise, the generated instance is deemed noisy. For example, in [109], words or phrases are replaced with their synonyms according to geometric distribution for text classification. [99] applies four simple data augmentation approaches based on synonym replacement and deletion. [97] applies pre-trained word embeddings to represent words in the form of high-dimensional vectors. Cosine similarity is used to find synonyns and finally the $K$ nearest words are selected as synonyms. Apart from static word embeddings, contextualised word embeddings are also used for text augmentation. For example, [46] first selects a target word and then predicts its replacement using a bi-directional language model and [101] adapts BERT to a conditional BERT for contextual augmentation. In addition to synonym replacement, other data augmentation approaches in NLP include back translation [86] that leverages English (with large available resources) to enrich training data for a less-resourced language and text generation that for example, GPT-2 [80, 96] is triggered to generate a supplementary sentence when being fed a given sentence in the original dataset as the conditional context prompts.

## 2.2 Text on Social Media

The use of text on social media for various purposes has been widely studied in the literature. Given its wide use in many problem domains, this section only examines those aspects of the work that are closely related to this research.

### 2.2.1 Text Cleaning

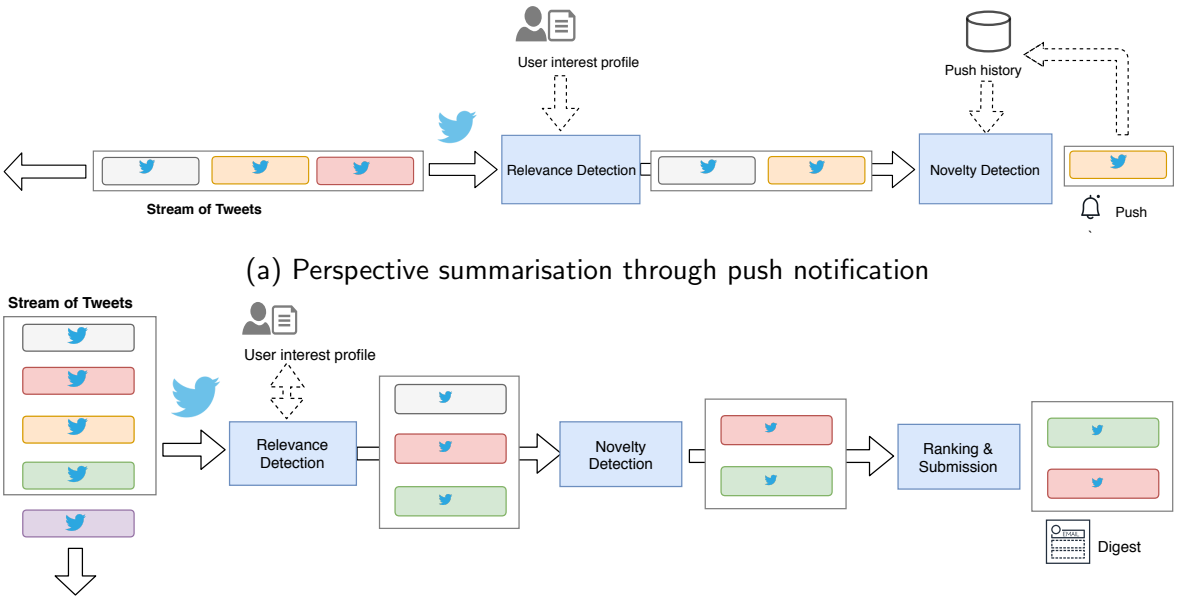
The messages that users post are heterogeneous, short[3], duplicated and noisy in one form or another, including abbreviations, misspellings, etc. Some methods have been developed to clean this kind of text, which act on two possible levels.

[3]On Twitter, a tweet is allowed to be no more than 280 characters in length.

**At word level.** Given a tweet, it is usually split into individual tokens using certain rules [72, 73, 25, 88]. One example of a tokeniser specifically designed for Twitter is TweetNLP[4]. If a system is specialised for one language, say English, non-English characters and symbols are usually removed [25]. For tweets containing special symbols such as hashtags, user mentions, or numbers, the commonly-practiced methods replace these with specialised tokens for vocabulary reduction. For example, in [76], a word embedding model is pre-trained on Twitter corpora by replacing all hashtags with $\langle hashtag \rangle$, user mentions with $\langle user \rangle$, and number with $\langle number \rangle$. In [88], a hashtag segmentation technique is applied and the hashtags are replaced with the segmented words. For handling misspellings or abbreviations, lookup dictionaries are commonly used. The dictionaries include, for example, the lexical normalization dictionaries [55] for correcting misspellings and Out Of Vocabulary (OOV) lookup dictionaries for correcting OOV words [38].

**At sequence level.** A tweet can be described as a sequence of text. Duplication is often seen among the sequences. At the cleaning stage, some techniques have been developed to filter out the duplicated or near-duplicated sequences so as to increase the efficiency of the systems that process them. The process of de-duplication measures the similarities between pairs of sequences. Similar sequences are grouped into clusters [29]. The similarity can be measured by simply looking the number of co-occur phrases in two sequences and then can be computed by Jaccard distance [91], KL-divergence [59], or cosine similarity [53]. More advanced methods go to grouping sequences by their semantic similarities [68]. Some practices in this category include topic-similar clustering through word2vec [66, 98] and language model through probability estimation [92].

### 2.2.2 Real Time Summarisation




(a) Perspective summarisation through push notification

(b) Retrospective summarisation through email digest

Figure 2.7: Real-time Summarisation on Twitter

One typical application of social media based on its real-time feature in information retrieval is real time summarisation (RTS) [71, 87]. The task takes a user's interest profile (describing the user's information need) as input to a filtering system that summarises a list of message streams in relevance to the profile and finally return matched ones to the user. These filtering systems can be classified into two types. Table 2.7 presents the architectures of the two types on Twitter, a) perspective summarisation through push notification and b) retrospective summarisation through

[4]http://www.cs.cmu.edu/ ark/TweetNLP/

email digest [87]. Both types emphasises the importance of information redundancy and thus a step called novelty detection is taken for de-duplication purposes before returning the summarised lists to users. Regarding the difference, the former processes tweets in real-time to help users keep up-to-date in a timely fashion whenever a new development occurs about their topics of interest. In contrast, the latter processes tweets in a batch and returns summarised lists to users in the form of email digests. Hence, for this type of system, latency is usually much less of a key factor. In the literature, perspective summarisation systems develop less computationally expensive techniques for relevance computation in consideration of latency, such as simple vector space model using TF-IDF [83, 92, 98], or word2vec weighting schemas [98]. In contrast, retrospective summarisation systems are usually equipped with more sophisticated algorithms, such as ensemble learning [68], hybrid filtering [92], etc. This is a good example of how the tradeoff between latency and accuracy is handled by an automatic filtering system.

### 2.2.3 Crisis Detection and Tracking

Crisis Detection and Tracking is a branch of Topic Detection and Tracking (TDT) that applies newswires as data streams for TDT [36]. Unlike newswires, content on social media is characterised by being short and unstructured. Considering this, this review only focuses on crisis-relevant detection and tracking.

Crisis detection reports important information of an abrupt crisis, such as the time and location of a terrorist attack. This information is closely associated with the responsive actions that emergency services agencies can take to help the people in need. A straightforward method for crisis detection is through burst detection that captures an incident by monitoring the increase ratio of tweets obtained by crisis-based keywords in real-time as compared to its records in history. For example, *TweetInfo* [62, 36] is a system developed for event detection and summarisation. It works first by extracting tweets with regard to a user-defined query and then detecting events by looking sudden increases in the ratio of tweets that contain the query as compared to the historical tweets that contain that the same query. The system can be used for various events and its application in crises like earthquake has demonstrated its effectiveness. Other systems for crisis detection in the literature include *Tweet4act* [18] and *TEDAS* [54].

Followed by crisis detection, crisis tracking is an activity to find important information about how a crisis evolves and unfolds. This type of information can be very helpful for emergency response agencies to deal with the aid needs from the people on the ground as a crisis unfolds. So far the major studies for crisis tracking have been to identify different phases of a crisis. For example, [39] presents a classifier for automatically categorising a crisis into three different phases: before, during, and after.

To respond to crisis more specifically however, a more fine-grained process is necessary. This motivates studies into the field of automatic classification for emergency response, which aims to identify specific users' needs in crisis tracking. The next chapter details the process of automatic classification for emergency response on social media, which is the main focus of this report.

# Chapter 3: **Emergency Response on Social Media**

This chapter discusses the main problem domain of this research. It involves the specific task of examining the state of the art in emergency response using tweets and other relevant data, aiming to fully answer RQ1. First, this chapter gives an introduction to the TREC Incident Streams (IS) track that defines the task of categorising users information needs in crisis-related tweets. This track is run annually and specifically designed for participants who are seeking computational techniques for emergency response. Not only does this track provide an annually-accumulated dataset (Section 3.1.1) that can be used as the basis of this research but also a set of evaluation metrics (Section 3.1.2) that are aimed at fairly evaluating the participating systems. Section 3.1.3 examines the methods that have been used for this track in past iterations of this track (and which helped to inspire the work discussed in Chapter 4). Although the IS dataset can serve to answer most of the research questions, given the scope of this research, only having it is insufficient. Section 3.2 presents other relevant datasets collected to meet different needs of this research, including unlabelled crisis data for unsupervised learning (Section 3.2.1) and crisis (in-domain) or general (out-of-domain) datasets for supervised classification (Section 3.2.2).

## 3.1 Incident Streams Track (IS Track)

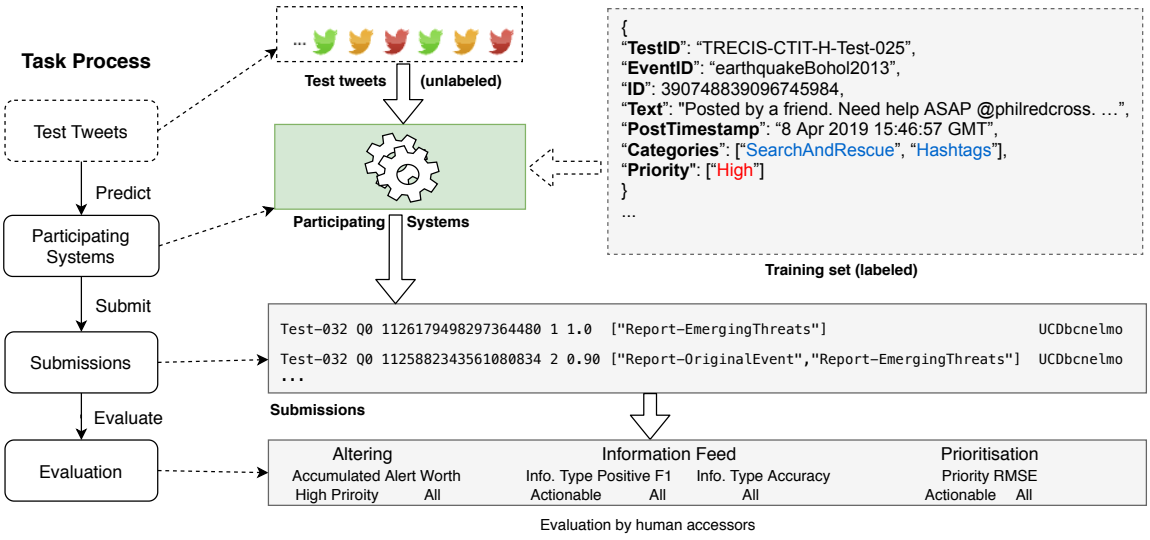



Figure 3.1: The task process of TREC Incident Streams (IS) track

The IS track is an annual task run as part of the Text REtrieval Conference (TREC)[1], aiming to encourage more mature social media-based emergency response technology. Figure 3.1 demonstrates the task process. Overall, the task is based on a dataset (known as the TREC-IS dataset introduced in Section 3.1.1) to find crisis-related tweets with actionable information. Unlabelled test tweets are fed into the participating systems. These are crisis-related tweets across different

[1] https://trec.nist.gov/

events. The participating systems are trained on the TREC-IS dataset to make predictions for the unseen tweets. Next, the predictions are formatted into submissions that are forwarded to the TREC organisers. Each test tweet is predicted with one or more aid types and a numeric priority score by which the predicted tweets are ranked in descending order. The submissions are subsequently judged by human assessors. There are different evaluation metrics in the returned results, which are defined to measure different aspects of the filtering systems' performance. The following sections introduce the TREC-IS dataset, the evaluation metrics, and primary methods used by participating groups respectively.

### 3.1.1 TREC-IS Dataset

The dataset are accumulated annually. Following the 2019 edition, there are around 35k tweets manually annotated according to information types (ITs) and priority. The ITs describe the aid types/needs with respect to a tweet that can be organised in a hierarchical structure. At a top level, it could be request, report, call to action, or other. At a low level, there are 25 ITs that imply more specific needs, and these ITs are divided into 2 categories: actionable and non-actionable. Table 3.1.1 presents all 25 ITs, of which 6 are defined as actionable and the remaining 19 as non-actionable. Apart from the ITs, the tweets are also annotated by priority that describes the urgency level of the tweets: critical, high, medium or low. Each tweet is assigned one or more ITs (multi-label across 25 classes) and only one priority level (single-label across 4 classes).

| Actionable | Non-actionable |
|---|---|
| Request-SearchAndRescue, | Other-ContextualInformation, Other-Advice, CallToAction-Donations |
| Request-GoodsServices, | Report-News, Other-Discussion, Report-CleanUp, Report-Factoid |
| CallToAction-MovePeople, | Report-FirstPartyObservation, Report-ThirdPartyObservation |
| Report-EmergingThreats, | Report-Hashtags, Request-InformationWanted, Report-OriginalEvent |
| Report-NewSubEvent, | Report-Official, Report-MultimediaShare, CallToAction-Volunteer |
| Report-ServiceAvailable | Other-Sentiment, Report-Weather, Report-Location, Other-Irrelevant |

Table 3.1: Actionable vs non-actionable information types in TREC-IS dataset.

### 3.1.2 Evaluation

There are three categories of evaluation metrics as in Figure 3.1, alerting, information feed, and prioritisation[2].

**Alerting (ranges from -1 to 1)**. The accumulated alert worth (AAW) is used to evaluate how effectively a system identifies critical and actionable tweets for alerting. AAW penalises systems that push too many false alerts, even if they also find many actionable tweets. For its sub metric, "High Priority", a tweet with priority score larger than 0.7 is considered to be an alerting tweet. Finding such tweets correctly results in a higher score in this metric.

**Information Feed (ranges from 0 to 1)** is used to test how effectively a system can categorise the tweets into the 25 ITs. Furthermore, the metric defines two sub-metrics that are used to distinguish between actionable and non-actionable ITs. F1 refers to the harmonic mean of precision and recall. Actionable IT F1 is the averaged F1 only across the actionable types as in Table 3.1.1, while All F1 refers to the averaged F1 across all ITs. To report noise, the overall classification accuracy is also included in this category.

[2]The metrics are defined at `http://dcs.gla.ac.uk/~richardm/TREC_IS/2019/TREC_IS_Metrics.pdf`

**Prioritisation (ranges from 0 to 1)** applies root mean squared error (RMSE) to evaluate how well a system estimates the importance score of tweets concerning priority. To quantify the priority levels, they are mapped to numeric score. Similarly to Information Feed, RMSE is calculated separately for only actionable and all ITs.

### 3.1.3 Methods Used

The IS track has been run since 2018, there are so far three runs from the track: 2018, 2019-A, and 2019-B (the track ran twice in 2019). After fully surveying the methods used by the participating groups, below is a summary of those participating methods that offer direct inspiration for the work presented in Chapter 4.

- **BJUT-2018** [58]. Given the small training set in 2018, this group enriched each IT by expanding it with supplementary relevant information from search engines or news sites. In the filtering system, they first preprocessed the tweets and then applied TF-IDF feature representation to train a SVM model for IT categorisation. The priority was estimated by a quantitative score of each IT associated with the corresponding priority levels in the training set.
- **CBNU-2018** [17]. This group mainly adopted two methods for the track. The first is a SVM model based on conceptual representation. Conceptual representation refers to tweets represented by terms, event entities, category indicator entities, and IT entities. The second is class activation mapping with one-shot learning in CNNs combining with the conceptual representation.
- **DLR_DW_BWS-2018** [49]. This group manually annotated extra tweets from existing crisis datasets such as *CrisisLexT26* and applied a data augmentation technique called automatic round-trip translation to expand the training data. With the expanded data, regarding model training for IT classification, the group applied different methods, including a logistic regression classifier on word statistics and a fusion CNN, etc.
- **NHK-2018** [67]. This group proposed label embedding using the hierarchical structure of labels for tweet classification. The group fed the input tweet text as well as all labels in a hierarchical structure to a LSTM-based network that is trained for predicting binary relevance between the tweet and each label. The method has been demonstrated to be effective for classification using imbalanced small datasets compared to several strong baselines.
- **CMUInformedia-2019-A** [42]. For IT prediction, this group exploited several features to represent the tweets first, including meta-information at different levels (tweet-level and word-level), user entity (i.e., user characterstics), and textual embeddings (i.e., GloVe, FastText, Skip-Thought, and BERT) and then trained an ensemble of ML classifiers (i.e., Naïve Bayes, Linear SVM, Random Forest, and XGBoost) on these features. For priority score prediction, they trained an estimator that combines the scores derived from the predicted ITs and the scores produced by a regression model.
- **DLR-2019-B** [48]. This team exploited word and sentence embeddings for classifying crisis-related tweets. This team is the same one as DLR_DW_BWS in 2018 and thus applied new methods this year. It proposes a deep neural network (DNN) with pre-trained BERT word embeddings and commonly-used sequence embeddings in literature. The approaches for generating sequence embeddings include smooth inverse frequency (SIF), Google's universal sentence encoder (USE), and mean-max attention autoencoder (Mean-Max AAE).
- **DICE_UPB-2019-B** [31]. In this group, several preprocessing steps were taken to clean tweets before model training, including emojis conversion, lemmatisation, stopword removal,

etc. After tweets are cleaned, then they fine-tuned BERT for multi-label tweet classification with two different loss functions, binary cross-entropy and focal loss.

- **IRIT-2019-B** [25]. This group extracted a large number of features before model training. The features are extracted from different perspectives, including user-based features, content-based features and even grammatical features (i.e., part-of-speech, name entities). The group adopted a combination of random forest and gradient tree boosting for IT prediction and a linear regression model for priority estimation. Moreover, to mitigate the imbalance class problem, SMOTE was applied to augment the dataset.

Given the methods as summarised, their implementations in the system are supported by some open and free frameworks or tools. For NLP tools, the choices include `NLTK` [8], `StandfordNLP` [78], `Flair` [2], `AllenNLP` [28], `Transformers` [100]. For model building, the commonly used machine learning framework is `scikit-learn` and deep learning frameworks include `Pytorch` or `TensorFlow`. For different types of text classification (multi-label, hierarchical multi-label, etc.), the open source library `NeuralClassifier` [56] is a good choice.

## 3.2 More Data Collections

Aside from the TREC-IS dataset, more data is required for the needs of this research. Data collection for unsupervised learning is necessary because one of the research questions is how to clean and enrich training crisis-related messages. For example, learning an OOVs dictionary may help cleaning noise (abbreviations or misspellings) within the messages, or pre-training a model may help get a good general linguistic knowledge of the messages before they go to classification models. All these studies rely on unsupervised learning that is based on unlabelled big crisis data. Regarding the datasets for supervised classification, they are collected for different purposes. For example, the crisis-related datasets can be used to explore text augmentation techniques (as listed in RQ2). The general datasets can be used as the benchmarking analysis of existing techniques such as word embeddings, or proposed techniques such the hierarchical multi-label classification model associated with RQ3. Given different research needs, below are the details of the datasets.

### 3.2.1 For Unsupervised Learning

The data for unsupervised learning mentioned here are mainly the unlabelled crisis tweets obtained from Twitter given its data availability. Twitter provides programmatic access to their content through APIs [3]. The data usually can be accessed in two forms: historic archived content and real-time/streaming content. The Twitter official API allows data collectors to obtain a random sample of 1% of all postings through its public streaming API. The data collected in this way is important due to the following concerns related to this research.

- **Pre-training**. For text, pre-training on big corpora helps gain a general knowledge of language. As discussed previously, the pre-training of word embeddings or Transformer-based language models has proven to be effective for downstream language tasks. It would be interesting to know the effectiveness of pre-training on the crisis-domain corpora. For this specific research, unlabelled big crisis data is needed. The data can be obtained through the Twitter APIs or by directly using some off-the-shelf corpora available online, such as

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

*CrisisNLP#1* [38] and *CrisisNLP#7* [4]. These tweets are provided with only tweet ids due to the limited right of direct distribution of raw tweets.

- **Crisis Lexicon or OOVs**. Similar to pre-training, unlabelled big crisis data used for summarising lexicons or OOVs in domain is important for downstream tasks. For example, if such lexicons in the crisis domain are well summarised, they can be used for feature extraction or misspelling correction before classification. To conduct studies on summarisation, the data as introduced in pre-training can be used directly or sampled by keywords using Twitter APIs. The existing lexicons [72] or OOVs [38] can serve as benchmarking targets.

### 3.2.2 For Supervised Classification

For supervised classification, both in-domain and out-of-domain labelled data are required according to different research needs.

#### 3.2.2.1 Crisis Datasets

With crisis datasets, the research can not only explore text augmentation techniques but also the performance difference between different events given the same classifier. In other words, it is an interesting place to study how well a classification model in crisis event A transfers to event B. Table 3.2 lists major existing datasets for classification in crisis domain [4]. These datasets not only offer an opportunity of maturing cross-event classification models but also benchmarking comparison in experiments. The following briefly introduces these datasets.

| Dataset | # classes | Type | Reference |
|---|---|---|---|
| CrisisLexT6 | 2 | binary | A. Olteanu, et al.[72] |
| CrisisLexT26 | mix. | mix. | A. Olteanu, et al.[73] |
| CrisisNLP#1 | 9 | multi-class | Muhammad Imran, et al.[38] |
| CrisisNLP#7 | 2 | binary | Firoj Alam, et al.[4] |

Table 3.2: Major existing datasets for classification in crisis domain.

- **CrisisLexT6** is originally composed of around 10 million tweets posted during 6 crisis events in 2012 and 2013, sampled by keywords, geographical regions or coordinates through the Twitter API. Around 60k (10,000 in each event) are labelled by crowdsourcing workers according to their relatedness ("on-topic", or "off-topic").
- **CrisisLexT26** consists of 28,000 tweets during 26 crisis events in 2012 and 2013 (around 1,000 in each event), labeled by crowdsourcing workers according to informativeness (informative or not informative), aid types (e.g. caution and advice, affected individuals, etc.), and information sources (e.g. media, NGOs, etc.).
- **CrisisNLP#1** consists of crisis-related tweets from three languages (English, French, and Spanish) collected during 19 natural and human-induced disasters. The samples are labelled by paid crowdsourcing workers according to 9 aid types (e.g. injured or dead people, donation needs, etc.)
- **CrisisNLP#7** contains around 21k tweets collected during two crisis events. The tweets are annotated by crowdsourcing workers according to relevance (relevant and non-relevant).

[4] Where mix. refers to multiple aspects of the dataset being labelled. The list is by no means comprehensive. For more crisis datasets, see `https://crisislex.org` and `https://crisisnlp.qcri.org`

#### 3.2.2.2 General Datasets

The collection of general datasets (i.e. out-of-domain) are out of consideration of several research purposes. First, these datasets are good candidates for studying existing techniques in language understanding. Because there are so many choices of pre-trained word embeddings (as introduced in Section 2.1.2.1), it is important to know which one suits what kind of dataset. As a result, it is necessary to conduct a comparative study using these general datasets for word embedding selection. The second purpose is that these datasets can be used for generalised testing. If a model (e.g., hierarchical multi-label classification model) is developed and performs well for crisis tweet classification, it is also desirable to test its performance on general/out-of-domain datasets. Below gives a list of varied general classification datasets.

| Dataset | # Classes | Type | Category | Reference |
|---|---|---|---|---|
| 20NewsGroup | 20 | multi-class | Topic | Lang, Ken, et al. [51] |
| SST-2 | 2 | binary | Sentiment | Socher, Richard, et al. [89] |
| AAPD | 54 | multi-label | Topic | Yang, Pengcheng et al. [102] |
| Reuters | 90 | multi-label | Topic | Apté, Chidanand et al. [5] |
| Patent | 9,162 | hierarchical multi-label | Topic | Wei Huang et al. [35] |

Table 3.3: Major existing benchmarking datasets for text classification.

- **20NewsGroup** dataset consists of around 18000 newsgroups posts on 20 topics. The dataset is originally split into training set (11314) and testing set (7532) based on messages posted before and after a specific date.
- **Stanford Sentiment Treebank (SST)** dataset contains phrases with fine-grained sentiment labels in movie reviews. It provides standard 5-way (SST-5) or binary (SST-2) classification splits for model training, development and evaluation.
- **arXiv Academic Paper (AAPD)** dataset is a multi-label dataset, which comprises paper abstracts extracted in the field of computer science through arXiv [5]. Each document is labeled with one or more subjects across 54 ones in total.
- **Reuters-21578 (Reuters)** is another benchmarking multi-label dataset for document classification, which consists of news content extracted from the Reuters newswire in 1987. In this dataset, each sample is labeled by one or more news categories out of 90 in total.
- **Patent** is a hierarchical multi-label dataset collected from the US Patent and Trademark Office[6]. It contains 100,000 US patents and each document contains textual information. There are 9,162 categories in total where each document can be in one or more of the categories (multi-label) and the categories are arranged by a four-level hierarchical structure.

## 3.3 Summary

This chapter first introduces the IS track that provides the TREC-IS dataset and standard evaluation metrics for this research, and summarises existing methods that are specifically applied to this track. In addition, the chapter introduces the collection of extra benchmarking datasets that can be used to support this research. The next chapter first reports the details work conducted so far by participating in the IS track. Then it discusses empirical work on word embeddings in text classification using some of the collected benchmarking datasets.

[5] https://arxiv.org/

[6] https://www.uspto.gov/

# Chapter 4: **Progress To Date**

To date, the first research question (RQ1) has been answered by conducting a detailed literature review, generally connected to text classification and social media (Chapter 2) and specifically related to emergency response (Chapter 3).

This chapter presents two pieces of work that contribute towards answering RQ2 and RQ3. Section 4.1 introduces the participation at 2019 TREC IS track (both edition A and B). Section 4.2 details a comparative analysis of word embeddings in classification, which focuses on providing insights for selecting appropriate word embedding in future participation in the IS track. The former explores several classification algorithms, data cleaning strategies and text augmentation approaches for the IS track. The latter work focuses on the study of pre-trained word embeddings used as the input representations in neural network based text classification, in order to find which are most appropriate to this domain. Some of this work has already been published, and two further papers emerging from work to date are in preparation, as detailed below:

- The work at IS track: **Congcong Wang**, David Lillis. *Classification for Crisis-Related Tweets Leveraging Word Embeddings and Data Augmentation*. In Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC 2019), Gaithersburg, MD, 2020.
- Word embedding analysis: In the process of proofreading and preparing for publication, targeting at CIKM 2020 (submission April 2020).
- Literature review: to be submitted to the International Conference on Pattern Recognition (ICPR 2020), Workshop on Research & Innovation for Secure Societies (RISS) (June 2020).

## 4.1 Participation at 2019 IS Track

The TREC IS track ran twice in 2019 (edition 2019-A and 2019-B). In edition A, the techniques used for classifying ITs and estimating priority mainly fall into traditional machine learning algorithms with hand-crafted features as input representations. After gaining some experience from the first participation, the participation at edition B aimed to explore the use of neural networks. Four runs were submitted in each edition and the evaluation results show that the runs outperform median for most situations and were the best in some metrics.

For both editions, the overall system architecture was as in Figure 4.1. The system began by preprocessing the tweets from a training set. The preprocessing was applied for all runs in both editions for data cleaning and refining. The preprocessing steps included converting all tweets to be lower-cased and removing URLs, numbers, punctuation and special characters (like #, &, ,etc.). In addition, an in-domain, OOV dictionary [38] was used to correct any misspelled words in tweets. Dataset analysis found that its classes were severely imbalanced (it contains few tweets with more urgent ITs and more critical priority levels), therefore data augmentation strategies are applied. The augmentation strategies were optional and varied across the submitted runs. After preprocessing and augmentation, the tweets were fed to train two models separately for IT classification and priority estimation. The models trained at this step also vary in the submitted runs. Once the models were trained, they were used to make predictions for the unlabelled tweets

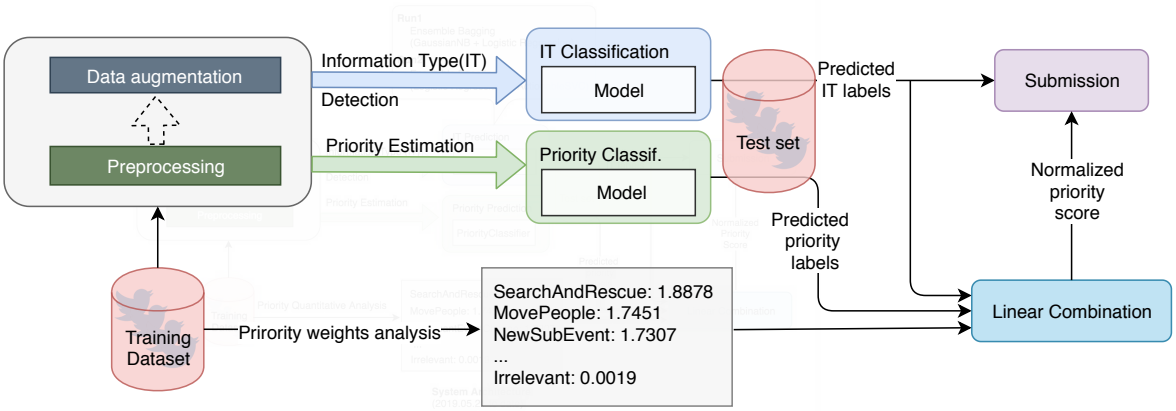



Figure 4.1: System architecture at TREC IS track

in the test set. At this stage, each test tweet was predicted with one or more ITs and one priority label. As the submission required the estimation of a tweet's priority with a score between 0 and 1, a linear combination was used to calculate this, whose formula is defined as follows:

$$s_i = (1 - \lambda) \times w_i + \lambda \times \delta(p_i) \tag{4.1}$$

where, given a tweet $i$, $s_i$ refers to its submitted priority score and $w_i$ refers to the average priority weight. Given the predicted ITs predicted for tweet $i$, $w_i$ is the averaged weight by looking up an information-type-to-weight table as presented in Figure 4.1. The look-up table is constructed by quantitatively analysing all tweets in the training set (the weight for an IT is averaged over the priority scores of the tweets that belong to the IT). In the equation, $\delta$ is the mapping function that maps the predicted priority label $p_i$ to the numeric priority score according to the mapping schema: low=0.25, medium=0.5, high=0.75 and critical=1.0. In the formula, $\lambda$ is a constant set to be 0.5 by default, to give equal contribution to the predicted priority label and analysed priority weight. Thus, priority is quantified to a score between 0 and 1.

### 4.1.1 2019 IS Edition-A

In TREC-IS 2019-A, four runs were submitted, namely UCDrunEL1, UCDrunEL1, UCDrunELFB3 and UCDrunELFB3. All the runs were implemented based on machine learning models using feature representations. In all the runs, SMOTE was leveraged for mitigating the imbalance dataset problem and binary relevance was used for IT (multi-label) classification. In addition, a linear regression was trained for predicting priority levels. The runs differed in how they represented tweets and trained their IT classifiers with different scenarios. Table 4.1 summarises the combinations of four scenarios for the runs. Below are detailed the four scenarios.

| | Simple ensemble | Actionable-specific |
|---|---|---|
| Word2vec | UCDrunEL1 | UCDrunEL2 |
| Feature boosting | UCDrunELFB3 | UCDrunELFB4 |

Table 4.1: Four runs at TREC-IS 2019-A

**Word2vec**. For runs with this scenario, the tweets are presented through word embedding using an in-domain pre-trained word2vec model [38]. To generate a tweet representation, the bag-of-

means (BOM) strategy was used, which averaged the vectors of all individual words.

**Feature boosting**. This scenario combined the pre-trained word2vec representing a tweet with 21 other hand-crafted features. These features were extracted based on the commonly-applied practices in the literature (see Section 2.1.1). The features chosen were number of hashtags (numeric), named Entity count (numeric), URL count (numeric), digit count (numeric), special character count (@, ! and ?, numeric), number of "special" verbs (e.g. trapped, stuck, move, etc.: dataset statistical analysis), sentiment polarity (categorical, -1, 0 or 1), tweet length (word_length, char_length, numeric), capital ratio (numeric) and retweet check (binary, 0 or 1).

**Simple ensemble**. In this scenario, two classifiers worked together to decide the ITs to be assigned to a tweet: a logistic regression (LR) classifier and a naïve bayes classifier. The ITs predicted for a tweet were determined by the two classifiers with an averaged probability threshold=0.50, checking whether an IT should be assigned.

**Actionable-specific**. Similar to simple ensemble, this scenario also applied two classifiers to decide the ITs to be assigned to a tweet. However, the difference is that the classifiers here were a LR classifier and an actionable classifier. The actionable classifier in the system was actually a linear SVM model that was trained only on the training tweets with actionable types (see Table 3.1.1). To assign ITs for a tweet, the LR classifier first predicted a general list of ITs and then the predicted list was forwarded to the actionable classifier that helped predict any actionable ITs which the LR classifier did not predict for the tweet.

### 4.1.2 2019 IS Edition-B

In TREC-IS 2019-B, four runs were submitted, namely UCDbaseline, UCDbilstmalpha, UCDbilstmbeta and UCDbcnelmo. Each used a neural network based deep learning algorithms (except for UCDbaseline, which reused UCDrunELFB3 with the new data). For the remaining three runs, different word embeddings and data augmentation strategies were applied to explore the influence they have on the performance. The following describes the technical variations of each run.

**UCDbilstmalpha**. To face the imbalanced classed, this run applied the text generation strategy GPT-2 [80] for data augmentation. This extracts rare/less proportional tweet samples with respect to ITs and priority in the training set and then uses the extracted samples as conditional samples for GPT-2 to generate synthetic samples. Table 4.1.2 gives an example of a generated sample by GPT-2 given a raw critical tweet from training set as the conditional sample.

| Conditional raw sample | Generated sample |
|---|---|
| LANDSLIDE!... road blocked in Costa Rica after M7.6 earthquake | 1 car submerged on a hillside and one car in water |

Table 4.2: Generation of a critical tweet by GPT-2

For representing tweet, this run used GloVe in the embedding layer and a standard biLSTM as the sequence encoder (see Figure 2.3). For IT classification, $BCELoss$ (see Section 2.1.2) was used as the objective function where $w_i$ is set up to 1 (i.e., no loss weight). For priority level classification, $MultiCELoss$ was used as the objective function.

**UCDbilstmbeta**. Unlike UCDbilstmalpha, this run did not apply data augmentation. Instead, $w_i$ in Equation $BCELoss$ was used for handling class imbalance and thus was not set to be 1 by default but $w_i = \frac{|l_{\max}|}{|l_i|}$ where $|l_i|$ is the number of samples of IT $l_i$ appearing in the training data and $|l_{\max}|$ is that of the most-frequent IT. In addition to this, this run also added a character-level embedding that is concatenated to the GloVe in the embedding layer.

**UCDbcnelmo**. The last run used a neural network architecture that has been recently shown to perform well on text classification tasks [63, 77]. Bi-attentive classification (BCN) was first introduced in [63]. BCN takes two sequences as input and applies a biLSTM encoder to create task-specific representations of each input sequence. One noticeable feature of the network is the application of a bi-attention mechanism. The attention mechanism is used to pay attention of each input representation to the other (so that one is conditional on the other). After the bi-attention process, conditional information of one on the other is obtained. BCN then again applies a biLSTM to integrate the conditional information before going to the output layer. In the output layer of BCN, a max-out network uses pooled features to compute a distribution over possible classes. In [77], BCN was combined with ELMo to achieve state-of-the-art performance in sentiment classification. BCN+ELMo applies the contextualised ELMo word representations in the embedding layer of BCN. Based on the promising performance that BCN+ELMo achieved in similar tasks, it was adapted to the crisis domain as the last run. In the embedding layer, there was not only ELMo but also GloVe. Like UCDbilstmalpha, this run also leveraged GPT-2 based data augmentation strategy for alleviating the imbalanced class problem.

### 4.1.3 Evaluation and Discussion

For evaluation, all participant groups submit their runs to TREC, and the track organisers employ human assessors to label all the submitted tweets. Then, the evaluation results are returned to all the participant groups. The results of all 8 runs are reported below. Results in bold are the best in their column. Apart from the presence of scores that the submitted runs achieved, the median scores for each metric are also included.

Table 4.3 presents the returned evaluation results for four runs at TREC-IS 2019-A. Generally, the runs in most metrics achieved performance above the median. In particular, actionable RMSEs for the runs are a lot lower than the median and the actionable F1 of UCDrunELFB3 is notably higher than the median (0.1556 is next after the best 0.1969). The weak aspect of the runs was in the information type accuracy, with no run above the median. This motivates more future work to optimising accuracy. To compare UCDrunELFB3 with UCDrunEL1, and UCDrunELFB4 with UCDrunEL2, it is shown that the use of feature boosting in UCDrunELFB3 and UCDrunELFB4 did not help in AAW but improved actionable F1 (0.1556 vs 0.1280 and 0.1177 vs 0.1070) and All F1 (0.1807 vs 0.1627 and 0.1617 vs 0.1467). Likewise, if we compare UCDrunEL2 with UCDrunEL1, the use of an actionable classifier in UCDrunEL2 did not help improve information type F1 but did improve both AAW and information type accuracy. In summary, the runs are some of the better performing ones in terms of information type categorisation, but are somewhat too conservative in terms of alerting. In contrast to a lot of other participating runs, the runs have a better distribution across classes (there are no complete class failures), likely due to the application of SMOTE for mitigating the imbalanced class problem.

Table 4.3: Evaluation results of four runs at TREC-IS 2019-A

| Runs | Alerting | | Information Feed | | | Prioritization | |
|---|---|---|---|---|---|---|---|
| | AAW | | IT Positive F1 | | IT Acc. | RMSE | |
| | High Priority | All | Actionable | All | All | Actionable | All |
| UCDrunEL1 | 0.8744 | -0.4442 | 0.1280 | 0.1627 | 0.7800 | 0.1105 | 0.0744 |
| UCDrunEL2 | **-0.8556** | **-0.4382** | 0.1070 | 0.1467 | 0.8151 | 0.1107 | 0.0793 |
| UCDrunELFB3 | -0.9343 | -0.4700 | **0.1556** | **0.1807** | 0.7787 | - | - |
| UCDrunELFB4 | -0.9268 | -0.4676 | 0.1177 | 0.1617 | 0.8258 | **0.1163** | **0.0656** |
| Median | -0.9493 | -0.4855 | 0.0459 | 0.15995 | **0.8539** | 0.1615 | 0.07545 |

Table 4.4 presents the returned evaluation results for four runs at TREC-IS 2019-B. Overall, the runs outperform the median for most situations. Figure 4.2 compares the runs among the top

13 participating runs out of 32 runs in total. All four runs fall in top 13 in both information feed and prioritisation. In particular, UCDbaseline achieves the best (better than any other participating runs) performance in finding actionable ITs as indicated by the actionable positive F1. Comparatively, UCDbilstmalpha is somewhat weak among the four runs. Although it hits an information type accuracy 0.86, which is marginally above the median of 0.8583, it does not do well in AAW, which indicates its weakness in correctly finding tweets for alerting. For UCDbilstmbeta, some of its improvements over UCDbilstmalpha are seen. In particular, UCDbilstmbeta achieves good performance in AAW, whose scores (-0.6047 and -0.3332) are increased from the median (-0.9197 and -0.4609) to some extent. It also outperforms UCDbilstmalpha in information type positive actionable F1. Its performance over UCDbilstmalpha implies that the use of character-level embedding and loss weights helps improve the overall performance. Compared to the other three runs, UCDbcnelmo obtained relatively even good performance across the metrics. It achieved better scores in almost every metric than the median to a good degree except that its information type accuracy is slightly lower than the median.

Table 4.4: Evaluation results of four runs at TREC-IS 2019-B

| | Alerting | | Information Feed | | | Prioritisation | |
|---|---|---|---|---|---|---|---|
| Runs | AAW | | IT Positive F1 | | IT Acc. | Priority RMSE | |
| | High Priority | All | Actionable | All | All | Actionable | All |
| UCDbaseline | -0.7856 | -0.4131 | **0.1355** | **0.1343** | 0.7495 | **0.0859** | **0.0668** |
| UCDbilstmalpha | -0.9287 | -0.4677 | 0.0614 | 0.1087 | **0.86** | 0.1521 | 0.0893 |
| UCDbilstmbeta | **-0.6047** | **-0.3332** | 0.1269 | 0.0607 | 0.8378 | 0.1004 | 0.0822 |
| UCDbcnelmo | -0.6961 | -0.3624 | 0.1099 | 0.1192 | 0.8452 | 0.1036 | 0.0769 |
| Median | -0.9197 | -0.4609 | 0.0386 | 0.1055 | 0.8583 | 0.1767 | 0.1028 |

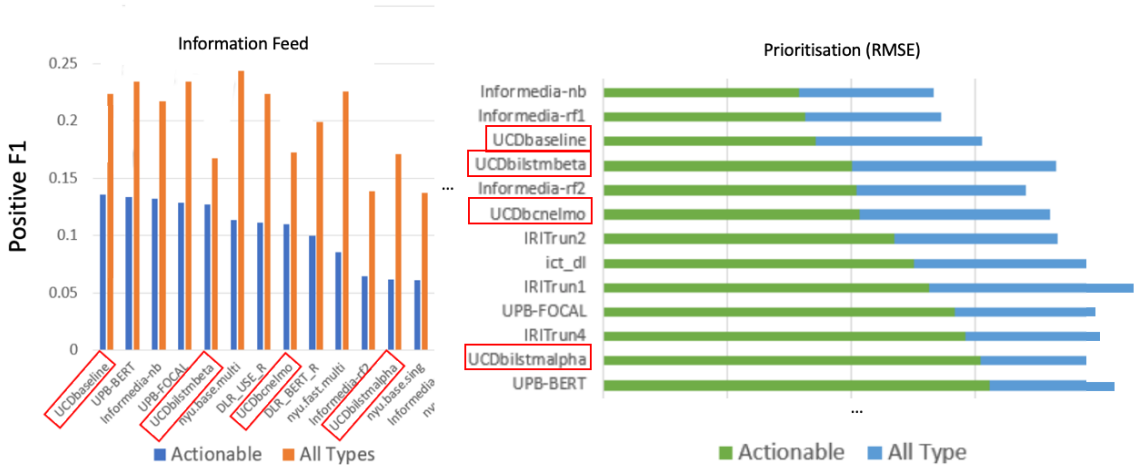



Figure 4.2: The runs at TREC-IS 2019-B among the top 13 participating runs out of 32 runs

## 4.2 Analysis of Word Embeddings for Classification

One question that arose from participating in TREC-IS 2019-B was what pre-trained embedding to choose, given that different choices of embedding layer perform differently on different data domains. This motivated a comparative analysis of widely-used word embeddings in neural network based classification, which is considered to be instructive for word embedding development in a specific problem domain, such as the emergency response domain.

### 4.2.1 Methodology

As so many different word embedding methods have developed, little work has done for systematically comparing them within a specific language task. This work constraints the specific language task to text classification. As a starting point, this study defines two encoders that encode word representations into sequence representations for classification in the network architecture as shown in Figure 2.3. The two sequence encoders are a simple CNN and biLSTM layer respectively. This work studied a wide range of pre-trained embeddings, from the classic word embeddings to more recent contextualised embeddings and report their impact on the performance in different classification domains. The primary motivation of the empirical study on word embeddings is not to compete with the state-of-the-art performance, but to present a methodology of experiment-oriented quick word embedding selection for text classification. The contributions of this work are summarised as follows.

- The work provides insights for choosing or developing word embeddings in a given classification domain using neural network based algorithms. The experimental methodology is transferable and applicable to other network architectures and language tasks.
- This work also presents an empirical survey on word embeddings. The survey is considered to be instructive for benchmarking analysis of word embeddings in neural network based text classification.
- This work produces a framework that can be used for efficient word embedding selection in neural network models given different text classification datasets (single- or multi-label).

### 4.2.2 Experimental Setup

The experiment setup includes the datasets selection, configurations for downstream models, and pre-trained embedding choices. The details of these components are given below.

**Datasets**. Four general datasets were selected, namely 20NewsGroup, SST-2, AAPD, and Reuters (see Table 3.3). These datasets vary in size, number of classes, types (single-label or multi-label), and categories (topic or sentiment). Figure 4.3 plots sample length distribution for the four datasets [1] and the following describes the experimental setup of each.

- 20NewsGroup. The train set is further split randomly into train set and development set at a ratio of 4:1.
- SST-2. The standard binary classification splits were used.
- AAPD. The experiment uses the splits provided by [1].
- Reuters. The standard ModApte splits [5] are used in the experiment.

**Model Training and Parameters**. The sequence encoders were a CNN and biLSTM layer in the neural network classification models. The configuration details of each are given below.

- CNN: an adapted simple version of the CNN model was used as the encoder. It is configured with the number of filters 100, and 3 filter sizes - 2,3,4 respectively.

[1] For better visualisation, only the samples with number of words less than 2000 and 1000 for 20NewsGroup and Reuters respectively are displayed. "std." denotes standard deviation where it is seen that 20NewsGroup comprises samples with the most variable length among the four datasets

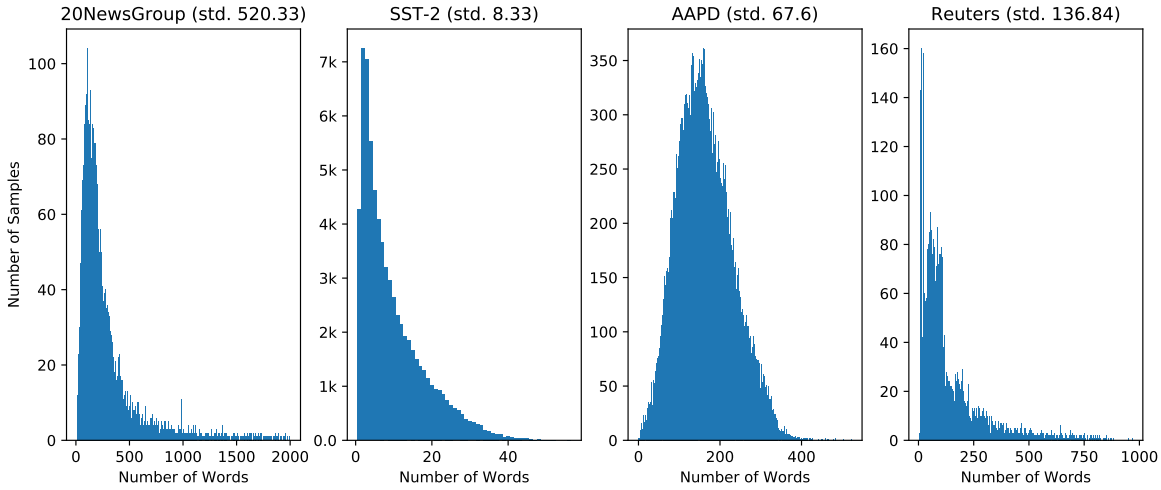



Figure 4.3: The number of words (sample length) versus the number of samples (sample frequency) of the datasets - 20NewsGroup, SST-2, AAPD and Reuters.

- LSTM: the simple biLSTM [113] is used as the encoder with hidden size being 256.

For common hyper-parameters, both models are trained with Adam optimiser (learning rate $=$ 0.001), in 140 training epochs, patience 5 and early stop based on accuracy evaluated on the development set. In addition, dropout is used with $p$ between 0.2 and 0.5 across layers of the networks to prevent over-fitting.

**Word Embeddings**. Given a neural network model, the word embeddings are used as the feature representations in the embedding layer. Below are described the word embeddings choices in the experiment.

- **Baseline**: The baseline simply defines a randomly-initialised trainable embedding layer with a dimension of 100 for encoding word inputs. The baseline is not pre-trained but learned from scratch with the other downstream model parameters. Hence, the impact that word embeddings have on performance when learned from scratch or pre-trained can be studied.
- **word2vec**: This uses the off-the-shelf 300 dimensional word2vec embedding pre-trained on Google News dataset.
- **GloVe**: Several pre-trained GloVe embeddings are available online. These embeddings vary in size, and being pre-trained in datasets including Wikipedia, Common Crawl, and Twitter. This experiment uses the 50-300 dimensional GloVe embedding pre-trained on Wikipedia (6B tokens) and 100 dimensional one pre-trained on Twitter (27B tokens).
- **FastText**: There are pre-trained FastText models available for many languages. Because the selected datasets are limited in English, the experiment only chooses the 300 dimensional English version of pre-trained FastText embedding.
- **Character-level embedding (Char)**: This defines a CNN encoder for learning character-level word embeddings from scratch. It includes the character embedding with the intention of exploring the performance difference when it is concatenated with the classic GloVe embedding as opposed to standalone FastText.
- **ELMo**: There are different sizes of pre-trained ELMo models that have been made available online. The original one is selected in the experiment. The linear weighted combination of the 3 layers of ELMo (i.e., character-based output, 1st LSTM output, 2nd LSTM output) is configured with one output representation using `AllenNLP`.

- **BERT**: The experiment includes BERT for studying its effectiveness in text classification when being used as a fixed feature extractor. The "bert-base-uncased" pre-trained variant was chosen to concatenate the last two layers for word representations. In the experiment, only SST-2 and Reuters are experimented on BERT as representatives of single-label and multi-label datasets respectively. Because BERT only accepts input sequences with the fewer than 510 tokens[2], a truncation method inspired by [90] was used, which only keeps the first 150 tokens of the sequences for Reuters considering its average sample length around 144.3 and resource constraints.

### 4.2.3 Results and Discussion

Based on the experimental setup described above, the evaluation results are averaged by repeating each experiment twice with different random seeds [22]. For all datasets, the results were evaluated on the test set accuracy. In addition to accuracy, the evaluation metrics also include macro-F1 for single-label datasets and micro-F1 for multi-label datasets.

Table 4.5: Accuracy and macro-F1 for single-label datasets (20NewsGroup and SST-2).

| | 20NewsGroup | | | | SST-2 | | | |
|---|---|---|---|---|---|---|---|---|
| | CNN | | biLSTM | | CNN | | biLSTM | |
| | Acc. | macro F1 | Acc. | macro F1 | Acc. | macro F1 | Acc. | macro F1 |
| Baseline | 0.7809 | 0.778 | 0.50315 | 0.5072 | 0.8138 | 0.8131 | 0.835 | 0.8349 |
| word2vec | 0.8237 | 0.81825 | 0.6173 | 0.6169 | 0.8287 | 0.8281 | 0.8432 | 0.8428 |
| G-50 | 0.7907 | 0.7854 | 0.6945 | 0.6941 | 0.8056 | 0.8052 | 0.8304 | 0.8300 |
| G-100 | 0.8095 | 0.8027 | 0.7189 | 0.7175 | 0.8177 | 0.8172 | 0.8383 | 0.8372 |
| G-200 | 0.8306 | 0.8243 | 0.7450 | 0.7421 | 0.8227 | 0.8222 | 0.8468 | 0.8465 |
| G-300 | 0.8251 | 0.8175 | 0.7352 | 0.7314 | 0.8166 | 0.8159 | 0.8515 | 0.8513 |
| G-T-100 | 0.7949 | 0.7901 | 0.6683 | 0.6668 | 0.8152 | 0.8150 | 0.8416 | 0.8415 |
| F-300 | 0.8226 | 0.8168 | 0.6391 | 0.6400 | 0.8174 | 0.8164 | 0.8449 | 0.8448 |
| F-300+G-300s | **0.8457** | **0.8375** | 0.7709 | 0.7677 | 0.8400 | 0.8399 | 0.8575 | 0.8574 |
| Char+G-100 | 0.8270 | 0.8194 | **0.7855** | **0.7804** | 0.8259 | 0.8252 | 0.8504 | 0.8502 |
| ELMo | 0.7922 | 0.7866 | 0.71805 | 0.7161 | 0.8811 | 0.8811 | **0.8954** | **0.8954** |
| BERT | - | - | - | - | **0.8855** | **0.88545** | 0.8866 | 0.8866 |

Table 4.5 and Table 4.6 present the evaluation results for single-label and multi-label datasets respectively [3]. Based on the evaluation results, several key findings are summarised as follows:

#### Non pre-trained vs pre-trained

When comparing the baseline with the remaining runs, the performance metrics indicate that improvements are seen when applying pre-trained embeddings, such as GloVe, FastText, ELMo, etc. This indicates that the embedding layer initialised with pre-trained embedding models enriches the feature representations of a sequence and thus results in better classification for the sentence. This result also coincides with the the observation in the literature that pre-trained word embeddings help neural network models perform well in general NLP tasks.

#### Model architecture selection: CNN vs biLSTM

[2] 510 is 512 subtracting the [CLS] and [SEP] tokens.

[3] The values in bold are the highest in columns. + denotes concatenation, G denotes GloVe (6B tokens), T denotes Twitter, F denotes FastText, and the appended number denotes the embedding dimension. Specifically, 300s refers to the GloVe embedding with 840B tokens pre-trained on Common Crawl. For example, G-T-100 means the off-the-shelf pre-trained *glove.twitter.27B.100*.

Table 4.6: Accuracy and micro-F1 for multi-label datasets (AAPD and Reuters).

| | AAPD | | | | Reuters | | | |
|---|---|---|---|---|---|---|---|---|
| | CNN | | biLSTM | | CNN | | biLSTM | |
| | Acc. | micro F1 | Acc. | micro F1 | Acc. | micro F1 | Acc. | micro F1 |
| Baseline | 0.351 | 0.6653 | 0.346 | 0.65045 | 0.74115 | 0.80915 | 0.5467 | 0.5996 |
| word2vec | 0.3585 | 0.6839 | 0.361 | 0.6761 | 0.8097 | 0.86535 | 0.78185 | 0.82725 |
| G-50 | 0.3425 | 0.66275 | 0.3485 | 0.67285 | 0.7973 | 0.8538 | 0.7883 | 0.83295 |
| G-100 | 0.3515 | 0.6808 | 0.3585 | 0.6797 | 0.8046 | 0.8589 | 0.7756 | 0.8307 |
| G-200 | 0.349 | 0.67065 | 0.3585 | 0.6791 | 0.81105 | 0.8628 | 0.7983 | 0.842 |
| G-300 | 0.357 | 0.6847 | 0.3545 | 0.6844 | 0.80625 | 0.8642 | 0.79915 | 0.8427 |
| G-T-100 | 0.352 | 0.67085 | 0.3465 | 0.66685 | 0.7958 | 0.8538 | 0.7741 | 0.8214 |
| F-300 | 0.3445 | 0.67325 | 0.345 | 0.6722 | 0.7902 | 0.84755 | 0.7512 | 0.79905 |
| F-300+G-300s | **0.363** | **0.6943** | **0.368** | **0.6876** | **0.81865** | **0.87105** | **0.8082** | **0.8496** |
| Char+G-100 | 0.349 | 0.67215 | 0.3615 | 0.6819 | 0.81385 | 0.86665 | 0.78305 | 0.83635 |
| ELMo | 0.3495 | 0.6739 | 0.3165 | 0.62445 | 0.8021 | 0.85795 | 0.74575 | 0.8014 |
| BERT | - | - | - | - | 0.8016 | 0.84805 | 0.6684 | 0.7391 |

In the study, CNN-based and biLSTM-based models are used for sequence classification. The results do not reveal a general advantage of one over another. Instead, the performance impact of model architectures depends on the dataset domain. For example, CNN has a clear advantage over biLSTM for 20NewsGroup but not for SST-2. For multi-label datasets, AAPD and Reuters, CNN has a slight advantage over biLSTM. This may be explained that biLSTM performs better in summarising a sequence representation whose length is small. Hence, the degraded performance that CNN achieved for SST-2 is likely because it is not good at capturing position dependencies between words of a sequence as opposed to biLSTM. However, as the sample length increases, biLSTM's performance drops significantly due to catastrophic forgetting. This explains why biLSTM hits a poor performance for dataset 20NewsGroup whose samples are lengthy. For model architecture selection, the same suggestion as [22] can be drawn: it is better to select pre-trained word embeddings first and then a model architecture based on dataset characteristics.

**Impact of different classic embeddings**

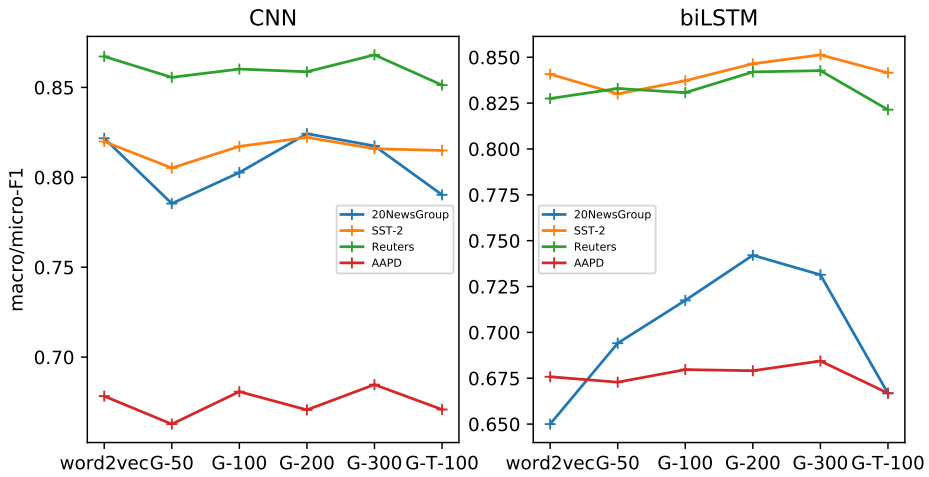



Figure 4.4: Embeddings (word2vec and GloVe) versus macro (20NewsGroup and SST-2) and micro (AAPD and Reuters) F1

In the study, word2vec, GloVe, and FastText are the three classic embeddings for comparative

analysis. To take a closer look at word2vec and GloVe variants, Figure 4.4 plots the performance difference between word2vec and GloVe variants in CNN or biLSTM models. It can be seen from the plots that overall GloVe has an advantage over the classic word2vec because its 100 dimension variant achieves almost the same performance as the 300 dimensional word2vec. For GloVe, the performance increases as the dimensions increase before 300 but drops gradually as it continues to increase to 300. In addition, its in-Twitter domain version does not achieve improved results on the four datasets. For selecting GloVe for a general classification dataset, it is good to use its 100-200 (6B tokens) dimensional pre-trained embeddings.

In addition to word2vec and GloVe, the experiment also studied FastText and character-level embedding. The two types of embeddings share the common feature that captures subword information. When using the 300-dimensional FastText as the standlone embedding, there is no evidence showing FastText outperforms GloVe. However, it achieves overall good performance when it is used in combination with GloVe. As seen in Table 4.5 and 4.6, the values in bold show that the embedding combination achieves the best performance for most situations. For character-level embedding, it was found that overall its concatenation with the 100 dimensional GloVe (Char+G-100) boosts the classification performance compared to the standalone use of GloVe (G-100). As a suggestion, it is worth considering GloVe as well as the character-level embedding to provide extra subword information.

**Classic embeddings vs contexualised embeddings**

Here the main finding captured from the evaluation results is that classic embeddings seem to have an advantage over contexualised embeddings for short sequence classification. Taking SST-2 as an example, it saw a significant improvement when using ELMo or BERT as opposed to GloVe, FastText, etc. However, the contexualised embeddings harm the performance in Reuters, where the metrics are similar to the baseline scores. Analytically, this may be attributed to the truncation for training samples of Reuters, leading to incomplete information of the samples before being fed to the embedding models. Moreover, it is also likely that contexualised embeddings work at a sentence level, which effectively embed a sentence with context information. For long, document-based datasets like Reuters, some better techniques are still required for representing a document given that its length is large before it is contextually embedded.

## 4.3 Modules Undertaken

The following is the academic record showing the modules undertaken, to satisfy the credits requirement of my Ph.D.:

| Module Code | Module Name | Credits | Semester | Academic Year | Result |
|---|---|---|---|---|---|
| SCI50030 | Academic Writing for Science for Postgraduate students | 5 | One | 2018/2019 | PX |
| COMP50040 | Online Research Skills | 5 | One | 2018/2019 | PX |
| COMP47600 | Text Analytics | 10 | One | 2018/2019 | B+ |
| COMP41380 | Communications and Outreach | 5 | One | 2019/2020 | PX |
| COMP30390 | Enterprise, Innovation, Entre. | 5 | One | 2019/2020 | A |

Table 4.7: Academic Credits Record

# Chapter 5: **Future Plan**

The previous chapters show that substantial progress has been made towards answering the research questions outlined in Section 1.3. This chapter discusses the plan to achieve those research questions that remain unanswered at this stage.

Figure 5.1 plots the Gantt chart demonstrating the timeline for the entire research project, which is divided into three work packages (WPs). The chart shows that work package 1 (WP1) has been accomplished, thus answering RQ1. This was achieved through the state of the art review, gathering and implementing existing approaches, and conducting extensive evaluation of these techniques to benchmark baseline methods.

The following sections describe the plan for completing the tasks in WP2 and WP3, some of which have been partially completed. The remaining tasks will result in the remaining research questions being answered by the completion of this PhD research.

## 5.1 Multi-label Classification

One core question of this research is to present a message (short crisis-related messages in this research) for multi-label classification (the aid types characterised by multi-label in this research). This work package primarily includes the tasks exploring techniques for multi-label classification. The breakdown of the tasks are as follows.

**Task 2.1: Research on feature extraction based machine learning techniques.**

Some work has been done into machine learning techniques using hand-crafted features. As discussed in Section 4.1, the evaluation results show that further work is necessary to optimise the previous work at IS 2019 Edition A. Since the IS track continues to run in 2020, the future work specific to the next participation of IS track is to balance the trade-off between accuracy and other metrics for the system. For example, the objective will be to extract more informative features and develop novel text augmentation strategies so as to improve the system's overall performance.

**Task 2.2: Development of contextualised embedding on big crisis data.**

Pre-training a contextualised embedding model gains a general representation of the domain knowledge, which is beneficial to the downstream multi-label classification task in emergency response. Although there are some existing well-developed contextualised embedding models, there is still a gap to develop a novel embedding model in the crisis-related tweet domain. For example, tweets can be better represented if the model can be developed by taking into account the misspellings, short context, or abbreviations of the tweets. Regarding the dataset for this research, the plan is to collect them from some existing collections of unlabelled crisis tweets (see Section 3.2.1). For experimental setup, the existing state-of-the-art embedding models pre-trained on the crisis corpora similar to the proposed model can be used as baselines and performance evaluation can be conducted on downstream multi-label classification tasks.

**Task 2.3: Research on deep hierarchical multi-label classification approach**

Considering that hierarchical relationships exist between aid categories, this task aims to explore novel deep hierarchical multi-label classification (HMLC) models for emergency response. The HMLC problem still remains a challenge in the literature so much work needs to be further done. One recent approach to this problem is an attention-based recurrent network approach for HMLC [35]. My plan is to take this work as a baseline and further explore new approaches to achieve improvement over the existing approaches. The study needs hierarchical multi-label datasets, so I will not only use the experimental datasets from the baseline but also the TREC-IS dataset. At this stage, I incubated the implementation idea inspired by an existing work. It is the hierarchical label embedding proposed by [67] that uses hierarchical structure of labels for twitter classification. One limitation of this work is that labels have to be enriched with additional descriptive information before they are embedded in a hierarchical structure. How to solve the limitation is the major focus of this task.

**Task 2.4: Development of ensemble learning approach.**

In a real-life emergency response system, it is usually the case that a model performs well in one aspect while suffers from another aspect. For example, a multi-label classifier can correctly find messages for alerting but in a way that pushes too many false alerts. Although such a system achieves high alerting score, it is untrustworthy for the end users. This task brings future work to explore ensemble learning approaches for achieving overall optimal performance. To be more specific, for example, the ensemble learning can go to train multiple classification models separately for actionable and non-actionable categorisation of information feed.

**Task 2.5: Investigation of temporal aspects.**

This research needs the investigation of temporal aspects given the necessity of real-time emergency response. The investigation is to not only find the existing standard temporal evaluation metrics but also offer guidance for the development of an adaptive system that is described in the next section.

## 5.2 Development and Evaluation of Adaptive System

The major research question that this work package aims to answer is the trade-off between efficiency and effectiveness, namely, latency and accuracy. Below is the breakdown of this package.

**Task 3.1: Research and development of adaptive system**

Different techniques are usually advantageous in different situations. In the real-world deployment of an emergency response system, the techniques should be applied as per specific needs, such as the temporal aspect, the explainability or the generalisation across crises. This motivates the development of an adaptive system that applies a specific technique out of a set of candidates to a specific situation. For example, feature extraction based machine learning approaches are necessary for situations where latency or explainability is an important requirement. However, neural network based deep models are necessary for situations where accuracy is highly strict. Another example is that some models achieves both high efficiency and effectiveness in certain crisis events while low in other crisis events. The adaptive system may also need to concern the adaption of techniques to different types of crises.

**Task 3.2: Evaluation of overall dynamic system**

According to the requirements of the adaptive system, the evaluation is to be conducted in three aspects to comprehensively measure its performance. The three aspects are effectiveness,

efficiency and generalisation. First, the effectiveness concerns how effective the system is at finding actionable messages for alerting or identifying messages according to priority levels. The metrics proposed in the TREC-IS track can be used to measure this aspect of performance. Secondly, the efficiency is concerned with how quickly the system can make a decision on an upcoming sample. There are several existing metrics to measure this aspect of performance, such as the inference time of a prediction for a sample. Lastly, generalisation is concerned with how well the system generalises from one crisis event to another. Future work will need to explore metrics to measure this aspect of performance.

**Task 3.3: TREC participation**

This participation in TREC offers the opportunity of further research on the classification of crisis-related tweets and priority estimation. If the IS track stops running in the long-term, the accumulated TREC-IS dataset can be used for the aforementioned research and IS evaluation metrics can be used to measure the system's performance in effectiveness. IS will run in 2020 and, as is customary, the full list of TREC tracks for 2021 will be announced later in the year.

**Task 3.4: Non-TREC dataset curation.**

Some work has been done to curate the non-TREC datasets (see Section 3.2). The non-TREC dataset curation will continue to support the adaptive system. The non-TREC datasets along with the IS dataset will be used as the benchmarking datasets in the aforementioned research, which will finally serve to the adaptive system.

## 5.3 Gantt Chart and Publications

Based on the aforementioned future plan, any research output targets at being published in the following conferences or journals (ranked by date distance to now):

- Conference on Empirical Methods in Natural Language Processing (EMNLP), submission deadline usually in **May** annually.
- Conference on Neural Information Processing Systems (NeurIPS), submission deadline usually in **May** yearly.
- Text Retrieval Conference (TREC), Incident Streams track, submission deadline usually in **Nov**. yearly.
- International Conference on Information Systems for Crisis Response and Management (ISCRAM), submission deadline usually in **Dec.** yearly.
- Special Interest Group on Information Retrieval (SIGIR), submission deadline usually in **Jan**. yearly.
- International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI), submission deadline usually in **Feb**. yearly.
- Annual Conference of the Association for Computational Linguistics (ACL), submission deadline varies yearly.

Figure 5.1: Gantt chart of this research

# Bibliography


[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *CoRR, abs/1904.08398*, 2019.

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland 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)*, pages 54–59, 2019.

[3] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[4] Firoj Alam, Shafiq Joty, and Muhammad Imran. Domain adaptation with adversarial training and graph embeddings. 2018.

[5] Chidanand Apté, Fred Damerau, and Sholom M Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[7] Ghazaleh Beigi, Xia Hu, Ross Maciejewski, and Huan Liu. An overview of sentiment analysis in social media and its applications in disaster relief. In *Sentiment analysis and ontology engineering*, pages 313–340. Springer, 2016.

[8] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. volume 5, pages 135–146, 2017.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[11] M. D. Buhmann, Prem Melville, Vikas Sindhwani, Novi Quadrianto, Wray L. Buntine, Luís Torgo, Xinhua Zhang, Peter Stone, Jan Struyf, Hendrik Blockeel, Kurt Driessens, Risto Miikkulainen, Eric Wiewiora, Jan Peters, Russ Tedrake, Nicholas Roy, Jun Morimoto, Peter A. Flach, and Johannes Fürnkranz. Random decision forests. In *Encyclopedia of Machine Learning*, 2010.

[12] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684, 2011.

[13] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[14] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. volume 16, pages 321–357, 2019.

[15] Everton Alvares Cherman, Maria Carolina Monard, and Jean Metz. Multi-label problem transformation methods: a case study. *CLEI Electron. J.*, 14, 2011.

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

[17] Wongyu Choi, Seung-Hyeon Jo, and Kyung-Soon Lee. Cbnu at trec 2018 incident streams track.

[18] Soudip Roy Chowdhury, Muhammad Imran, Muhammad Rizwan Asghar, Sihem Amer-Yahia, and Carlos Castillo. Tweet4act: Using incident-specific profiles for classifying crisis-related messages. In *ISCRAM*, 2013.

[19] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, 2017.

[20] David Roxbee Cox. *Analysis of binary data*. Routledge, 2018.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[22] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. A comparative study of word embeddings for reading comprehension. *CoRR, abs/1703.00993*, 2017.

[23] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[24] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[25] Alexis Dusart, Gilles Hubert, and Karen Pinel-Sauvagnat. Irit at trec 2019: Incident streams and complex answer retrieval tracks. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC 2019)*, 2020.

[26] Mica R Endsley. Toward a theory of situation awareness in dynamic systems. In *Situational Awareness*, pages 9–42. Routledge, 2017.

[27] Julia Daisy Fraustino, Brooke Liu, and Yan Jin. Social media use during disasters: a review of the knowledge base and gaps. 2012.

[28] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[29] Alan Griffiths, H Claire Luckhurst, and Peter Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37(1):3–11, 1986.

[30] Debarati Guha-Sapir, Femke Vos, Regina Below, and Sylvain Ponserre. Annual disaster statistical review. 2011.

[31] Rricha Jalota Hamada M. Zahera, Ibrahim Elgendy and Mohamed Ahmed Sherif. Fine-tuned bert model for multi-label tweetsclassification. 2020.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. volume 9, pages 1735–1780, 1997.

[33] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[34] Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: Improving classification performance when training data is imbalanced. In *2009 second international workshop on computer science and engineering*, volume 2, pages 13–17. IEEE, 2009.

[35] Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. Hierarchical multi-label text classification: An attention-based recurrent network approach. In *Proceedings of the 28th ACM CIKM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, CHINA, Nov 3-7, 2019*, pages 1051–1060, 2019.

[36] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. volume 47, page 67. ACM, 2015.

[37] Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Extracting information nuggets from disaster-related messages in social media. In *Iscram*, 2013.

[38] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. *Twitter as a Lifeline: Human-Annotated Twitter Corpora for NLP of Crisis-Related Messages*. LREC, 2016.

[39] Akshaya Iyengar, Timothy W. Finin, and Anupam Joshi. Content-based prediction of temporal boundaries for events in twitter. *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, pages 186–191, 2011.

[40] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *Schedae Informaticae*, 25:49–59, 2016.

[41] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, 2017.

[42] Po-yao Huang Alexander Hauptmann Junpei Zhou, Xinyu Wang. Cmu-informedia at trec 2019 incident streams track. 2020.

[43] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[44] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR, abs/1412.6980*, 2014.

[46] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[47] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[48] Anna Kruspe, Jens Kersten, and Friederike Klan. Classification of incident-related tweets: Exploiting word and sentence embeddings. 2020.

[49] Anna Kruspe, Jens Kersten, Matti Wiegmann, Benno Stein, and Friederike Klan. Classification of incident-related tweets: Tackling imbalanced training data using hybrid cnns and translation-based data augmentation.

[50] Laura Lambert, Christos JP Moschovitis, Hilary W Poole, and Chris Woodford. *The internet: a historical encyclopedia*, volume 2. ABC-CLIO, 2005.

[51] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[53] Baoli Li and Liping Han. Distance weighted cosine similarity measure for text classification. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 611–618. Springer, 2013.

[54] Rui Li, Kin Hou Lei, Ravi V. Khadiwala, and Kevin Chen-Chuan Chang. Tedas: A twitter-based event detection and analysis system. *2012 IEEE 28th International Conference on Data Engineering*, pages 1273–1276, 2012.

[55] Fei Liu, Fuliang Weng, and Xiao Jiang. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044. Association for Computational Linguistics, 2012.

[56] Liqun Liu, Funan Mu, Pengyu Li, Xin Mu, Jing Tang, Xingsheng Ai, Ran Fu, Lifeng Wang, and Xing Zhou. Neuralclassifier: An open-source neural hierarchical multi-label text classification toolkit. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–92, 2019.

[57] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *CoRR, abs/1907.11692*, 2019.

[58] Ning Lu, Hesong Wang, and Zhen Yang. Bjut at trec 2018: Incident streams track. In *TREC*, 2018.

[59] Feifan Fan Yue Fei Chao Lv, Lili Yao Jianwu Yang, and Dongyan Zhao. Pkuicst at trec 2015 microblog track: Query-biased adaptive filtering in real-time microblog stream. 2015.

[60] David M Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics, 1995.

[61] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, 2001.

[62] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Rob Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *CHI*, 2011.

[63] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[64] Richard McCreadie, Cody Buntain, and Ian Soboroff. TREC Incident Streams: Finding Actionable Information on Social Media. In *16th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, pages 691–705, 2019.

[65] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

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

[67] Taro Miyazaki, Kiminobu Makino, Yuka Takei, Hiroki Okamoto, and Jun Goto. Label embedding using hierarchical structure of labels for twitter classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6318–6323, 2019.

[68] Bilel Moulahi, Lamjed Ben Jabeur, Abdelhamid Chellal, Thomas Palmer, Lynda Tamine, Mohand Boughanem, Karen Pinel-Sauvagnat, and Gilles Hubert. Irit at trec real time summarization 2016. 2016.

[69] Venkata Kishore Neppalli, Cornelia Caragea, and Doina Caragea. Deep neural networks versus naive bayes classifiers for identifying informative tweets during disasters. In *ISCRAM*, 2018.

[70] Fran H Norris. *Methods for disaster mental health research*. Guilford Press, 2006.

[71] Andrei Olariu. Efficient online summarization of microblogging streams. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 236–240, 2014.

[72] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[73] Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 994–1009, 2015.

[74] Leysia Palen and Sophia B Liu. Citizen communications in crisis: anticipating a future of ict-supported public participation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 727–736. ACM, 2007.

[75] Egon S Pearson. Bayes' theorem, examined in the light of experimental sampling. *Biometrika*, pages 388–442, 1925.

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

[77] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[78] Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[79] Zhaowei Qu, Xiaomin Song, Shuqiang Zheng, Xiaoru Wang, Xiaohui Song, and Zuquan Li. Improved bayes method based on tf-idf feature and grade factor feature for chinese information classification. *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 677–680, 2018.

[80] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[81] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85:333–359, 2011.

[82] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR, abs/1609.04747*, 2016.

[83] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[84] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Advances in Neural Information Processing Systems 33 (NIPS)*. 2019.

[85] Miriam Seoane Santos, Pedro Henriques Abreu, Pedro J García-Laencina, Adélia Simão, and Armando Carvalho. A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. *Journal of biomedical informatics*, 58:49–59, 2015.

[86] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, 2016.

[87] Royal Sequiera, Luchen Tan, and Jimmy Lin. Overview of the trec 2018 real-time summarization track. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018). Gaithersburg, Maryland*, 2018.

[88] Umme Aymun Siddiqua, Abu Nowshed Chy, and Masaki Aono. Stance detection on microblog focusing on syntactic tree representation. In *International Conference on Data Mining and Big Data*, pages 478–490. Springer, 2018.

[89] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[90] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *Chinese Computational Linguistics*, page 194–206, 2019.

[91] Reem Suwaileh and Tamer Elsayed. Exploiting live feedback for tweet real-time push notifications. In *TREC*, 2017.

[92] Jizhi Tang, Chao Lv, Lili Yao, and Dongyan Zhao. Pkuicst at trec 2017 real-time summarization track: Push notifications and email digest. In *TREC*, 2017.

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[94] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.

[95] Sarah Elizabeth Vieweg. *Situational awareness in mass emergency: A behavioral and linguistic analysis of microblogged communications*. PhD thesis, University of Colorado at Boulder, 2012.

[96] Congcong Wang and David Lillis. Classification for Crisis-Related Tweets Leveraging Word Embeddings and Data Augmentation. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC 2019)*, Gaithersburg, MD, 2020.

[97] William Yang Wang and Diyi Yang. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, 2015.

[98] Xiao Wang, Pengcheng Fan, Liang Cheng, Guoliang Xing, Zhihua Yu, and Xueqi Cheng. Ictnet at trec 2017 real-time summarization track. In *TREC*, 2017.

[99] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, 2019.

[100] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR, abs/1910.03771*, 2019.

[101] Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer, 2019.

[102] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. Sgm: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926, 2018.

[103] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc., 2019.

[104] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics, 2016.

[105] Kiran Zahra, Muhammad Imran, and Frank O Ostermann. Automatic identification of eyewitness messages on twitter during disasters. *Information processing & management*, 57(1):102107, 2020.

[106] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[107] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12:191–202, 2017.

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

[109] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[110] Ye Zhang and Byron C Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263, 2017.

[111] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A c-lstm neural network for text classification. In *CoRR, abs/1511.08630*, 2015.

[112] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495. The COLING 2016 Organizing Committee, 2016.

[113] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.