

Classification of Short-Texts Generated During Disasters: Traditional and Deep learning Approach

Shamik Kundu

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



Department of Computer Science and Engineering

June 2018

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Shamik Kundu.

(Signature)


(Shamik Kundu)


CS16MTECH11015


(Roll No.)

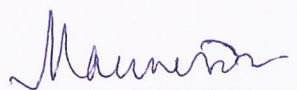
Approval Sheet

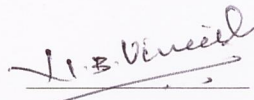
This Thesis entitled Classification of Short-Texts Generated During Disasters: Traditional and Deep learning Approach by Shamik Kundu is approved for the degree of Master of Technology from IIT Hyderabad


(Bheemardana Reddy T.) Examiner
Dept. of Computer Science and Engineering
IITH


(Manish Singh.) Examiner
Dept. Computer Science and Engineering
IITH


(Dr. Srijith P.K) Adviser
Dept. of Computer Science and Engineering
IITH


(Dr. Maunendra Sankar Desarkar) Co-Adviser
Department of Computer Science and Engineering
IITH


(Dr Vineeth N B) Chairman
Dept. of Computer Science and Engineering
IITH

Acknowledgements

To start with, I would like to thank my guides, Dr. Srijith P.K and Dr. Maunendra Sankar Desarkar for their valuable suggestions. Their constant guidance, patience, and immense knowledge were very helpful throughout the progress of my thesis. Next, I would like to express my gratitude towards the Computer Science and Engineering department at IIT Hyderabad for providing the motivation and resources to help me in successfully completing my work. I am also thankful to my seniors and friends for their friendly advice and words of encouragement in due course of my research.

Abstract

Micro-blogging sites provide a wealth of resources during disaster events in the form of short texts. Correct classification of those short texts into various actionable classes can be of great help in shaping the means to rescue people in disaster-affected places. The process of classification of short texts poses a challenging problem because the texts are usually short and very noisy and finding good features that can distinguish these texts into different classes is time consuming, tedious and often requires a lot of domain knowledge. In this thesis, we explore various non-deep learning and deep learning methods and propose a deep learning based model to classify tweets into different actionable classes such as resource need and availability, activities of various NGO etc. The proposed model requires no domain knowledge and can be used in any disaster scenario with little to no modification.

Keywords: Text classification, Topic Modelling, LDA, Word-embeddings, LSTM, Deep Learning

Contents

| | |
|--|------------|
| Declaration | ii |
| Approval Sheet | iii |
| Acknowledgements | iv |
| Abstract | v |
| Nomenclature | vii |
| 1 Introduction | 1 |
| 1.1 Overview | 1 |
| 1.2 Problem Definition | 2 |
| 1.3 Related Work | 2 |
| 2 Non-Deep Learning Approach | 3 |
| 2.1 Bootstrapping Approach | 3 |
| 2.1.1 Motivation | 3 |
| 2.1.2 Model Details | 3 |
| 2.2 Topic Modelling Approach | 4 |
| 2.2.1 Motivation | 4 |
| 2.2.2 LDA | 5 |
| 2.2.3 Model details: | 5 |
| 2.3 NLP Feature based Approach | 6 |
| 2.3.1 Motivation | 6 |
| 2.3.2 Model details: | 6 |
| 2.4 Results: | 6 |
| 3 Deep Learning Approach | 8 |
| 3.1 Motivation | 8 |
| 3.2 Word-embeddings | 8 |
| 3.2.1 Word2Vec | 8 |
| 3.2.2 GloVe | 9 |
| 3.2.3 FastText | 9 |
| 3.3 CNN Model | 9 |
| 3.3.1 Model details: | 10 |
| 3.3.2 Results: | 10 |
| 3.4 LSTM Model | 11 |

| | | |
|----------|---|-----------|
| 3.4.1 | Model details: | 13 |
| 3.4.2 | Results: | 13 |
| 3.5 | GRU Model | 13 |
| 3.5.1 | Model Details: | 15 |
| 3.5.2 | Results: | 15 |
| 4 | Experimental Details | 17 |
| 4.1 | Dataset Details: | 17 |
| 4.2 | Preprocessing: | 17 |
| 5 | Results and Comparison | 20 |
| 5.1 | Details of State-of-the-Art methods considered for comparison | 20 |
| 5.2 | Comparison | 20 |
| 5.3 | Analysis of Results | 20 |
| 6 | Conclusion and Future Work | 27 |
| | References | 28 |

Chapter 1

Introduction

1.1 Overview

In the modern era of technology, people use social media more than ever. Twitter or similar micro-blogging sites have become extremely popular for short daily updates. It has been observed that during any natural calamity, people tend to post about the situation in their local area [1]. Such kind of information is very useful resource for government or Non-Governmental Organizations (NGOs) for taking quick actions. During natural disasters like earthquake, flood, hurricane, tornado, etc. the means of physical communication such as roads, bridges etc often get damaged. It creates huge difficulties for the authorities to gather information about the situation at the disaster affected areas. However, this doesn't stop people at the core of affected areas to post updates in Twitter or similar sites about their location and current situation as long as power and network communications are still active in the regions. So if these tweets can be classified automatically into various categories such as resource needed or available, damage at a location etc, it will help government to decide the course of action needed and mitigate the pain and suffering of people at various affected areas.

Even though Twitter has such huge potential in lessening the severe effect of natural disasters on human lives, not many works have been done to exploit this. The main challenge in finding actionable insights from Twitter by classifying tweets into separate actionable classes is that the data is often very noisy. Different people use different words to provide the same information. Also there is a lot of mixing of regional languages along with English. The very short length of tweets doesn't help in classification either.

In this thesis, we work on the problem of mapping tweets posted disaster scenarios to several pre-defined action classes such as infrastructure damage, resources needed, resources available etc using various non-deep learning and deep-learning approaches. Our experiments show that semantic similarity is one of the most important deciding factors while classifying such short and noisy data. The performance of our best performing method (Deep LSTM Architecture) is compared against that of several other methods from literature. The empirical result indicate the effectiveness of the proposed method in classifying the tweets to appropriate action classes.

1.2 Problem Definition

The main goal is to classify texts (tweets in this case) into their respective classes. A single data point can belong to multiple classes making this essentially a multi-label multi-class problem. The problem can be formalized as follows: Given a set of data points $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, a set of class labels $C = \{c_1, c_2, \dots, c_m\}$, and their corresponding mappings, predict for each new datapoint \mathbf{x}_i , appropriate class label(s). Each \mathbf{x}_i is assumed to consist of l words, with t^{th} word in i^{th} example denoted by $\mathbf{x}_i^{<t>}$

1.3 Related Work

Application of tf-idf based models on disaster data: Olteanu et al. [1] used bag-of-words based approaches to classify micro-blogs either as relevant or irrelevant with respect to a particular disaster. Singla et al. [2] used seed keywords along with WordNet to retrieve disaster related tweets. Stowe et al. [3] classified tweets related to Sandy Hurricane into classes like sentiment, action, presentation etc using unigrams, POS Tags, Named Entities, URL, retweet information etc. as features and SVM and Naive Bayes as classifiers identifying actionable insights. Caragea et al. [4] used feature extraction technique to create a disaster-information retrieval system named Enhanced Messaging for the Emergency Response Sector (EMERSE). Fuji et al.[5] used a concept called inverse class frequency(ICF). ICF is same as inverse document frequency(IDF) but applied over classes instead of documents. They showed that this approach performs better in classification task. Ghosh et al. [10] proposed various tf-idf boosting methods to make use of the distinguishing power of the few specific terms.

Application of manual feature engineering in classification: Agarwa et al. [11]used various Part Of Speech(POS)-specific prior polarity features for classifying tweets into positive, negative and neutral classes. Anirban Sen et al. [12] argued that bag-of-words related features have severe limitation in cross domain application because of its over-dependency on vocabulary and used various POS based syntactic features to classify tweets with situational awareness which performed better than bag of words based models in cross-domain scenarios.

Application of deep learning based models on disaster data: To overcome the limitation of traditional approaches due to short and noisy data, There have been few works done recently that use deep neural networks for classifying tweets.

For example, Severyn et al. [6] used CNN for sentiment analysis of tweets. Wang et al. [7] used word embeddings with LSTM for identifying the polarity of tweets. Ma et al. [8] used RNN for detecting rumors from microblogs. Prany Khosla et al. [9] used attention based model for finding 'need' and 'available' tweets.

Chapter 2

Non-Deep Learning Approach

2.1 Bootstrapping Approach

2.1.1 Motivation

Since the research works in using tweets for disaster management have started only very recently, most of the data is unlabelled. The FIRE2016 dataset that is used in our experiment contains 95% unlabelled data. Semi-supervised approach can be helpful in exploiting the unlabelled data. We separate a part of labelled data for testing and remaining labelled data along with unlabelled data are used for training.

2.1.2 Model Details

The basic work flow of the model is explained below.

1. Initial Bag of words is generated manually for 7 given classes.
2. Initially, SVM is trained with labelled tweets.
3. After that, in each iteration, for a set of 1000 unlabelled tweets, the trained-SVM calculates probability of tweet belonging to each class.
4. Classification is done on the basis of Bag Of Words Classification and SVM Classification, giving weightage to each across different iterations.
5. The Bag of Words are continuously getting updated (improved) based on the recently classified tweets.
6. In each iteration, all classified tweets till that point are used for training SVM.
7. Since the Training data of SVM is increasing, the weight assigned to its score is also being increased across iterations.
8. Finally, after 5 iterations of training the SVM and updation of Bag of Words (increasing by 1000 tweets each time), test tweets are classified.

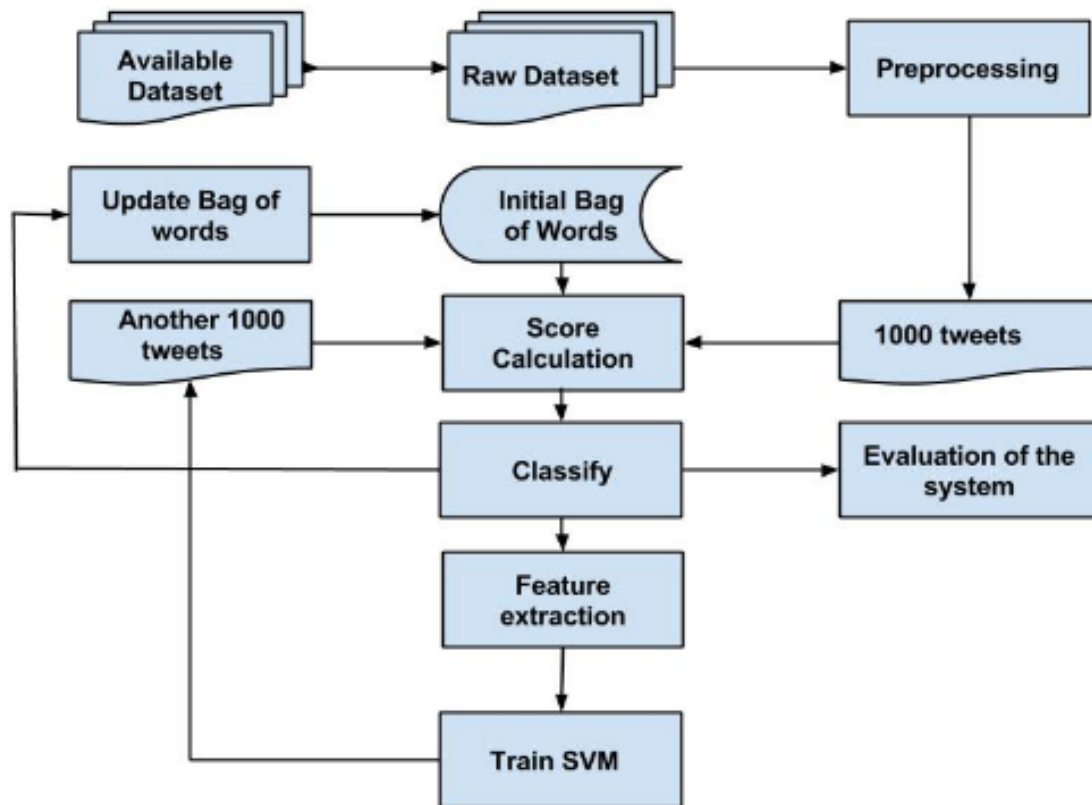


Figure 2.1: Block diagram of Bootstrapping approach

2.2 Topic Modelling Approach

2.2.1 Motivation

Tf-idf or BOW based approaches disregard valuable knowledge that could be inferred by considering the different types of relations between the words. These major relations are actually the essential components that, at a higher level, could express concepts or explain the main topic of a text. A representation method which could add some kind of relations and dependencies to the raw information items, and illustrate the characteristics of a text at different conceptual levels, could play an important role in knowledge extraction. The main motivation behind trying LDA as a part of our classification process is as follows:

1. LDA uses the global information of the whole corpus to extract the latent features for each document. Thus these features reflect more or less, the distribution of the whole corpus.
2. Since the number of topics are typically smaller than the vocabulary size, LDA representation can effectively help in reducing dimension.

Latent Dirichlet Allocation(LDA) is a probabilistic generative model for collections of discrete data and one of the simplest technique for topic modelling. There are a few variants of LDA such as supervised LDA, Labelled LDA, Phrase LDA etc.

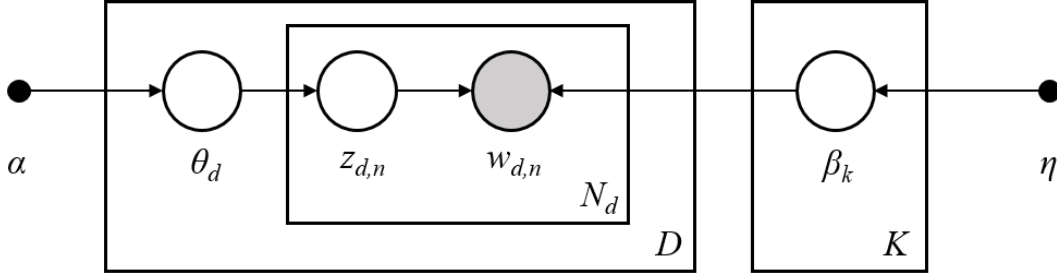


Figure 2.2: Directed Graphical Model of LDA

2.2.2 LDA

The basic terminologies used in LDA are as follows: A word is the basic unit of discrete data, defined to be an item from a vocabulary indexed by $1, \dots, V$. Words are represented using vectors with a single element equal to one and rest equal to zero which is also known as one hot encoding. A document is a sequence of N words denoted by $w = w_1, w_2, \dots, w_N$, where w_n is the n -th word in the sequence. A corpus is a collection of M documents denoted by $D = w_1, \dots, w_M$

The generative process of a vanilla LDA can be seen in the directed graphical model of Figure 1.1. It can be formulated using the following distributions:

1. Topic β_k : K topics drawn from a Dirichlet(η)

For each document d :

1. Topic proportion θ_d for each document drawn from Dirichlet(α)

For each document w in a document:

1. Topic assignment $z_{d,n}$ for each word in the document conditioned on topic assignment $\sim p(z_{d,n} | \theta_d)$
2. The observed variable w_n conditioned on the topic z_n and topics $\beta_k \sim p(w_{d,n} | z_{d,n}, k)$

The joint distribution of all the hidden and observed variables according the model can be written as:

$$\left(\prod_{k=1}^K p(\beta_k | \eta) \right) \left(\prod_{d=1}^D p(\theta_d | \alpha) \right) \left(\prod_{n=1}^N p(w_n | z_n, k) p(w_{d,n} | z_{d,n}, k) \right) \quad (2.1)$$

2.2.3 Model details:

Vanilla LDA and supervised LDA are used in this experiment. The main difference of supervised latent Dirichlet allocation (sLDA), with basic LDA is that in addition to the variables used in LDA, a response variable connected to each document is also used in sLDA. We jointly model the documents and the responses, in order to find latent topics that will best predict the response variables for future unlabeled documents.

For Vanilla LDA, 4 different set of number of topics are chosen such as 10,20,30 and 40. The output of 10-LDA, 20-LDA, 30-LDA and 40-LDA are then combined to form a vector of dimension 100 for each of the tweets. This vector is used as features to train a linear SVM.

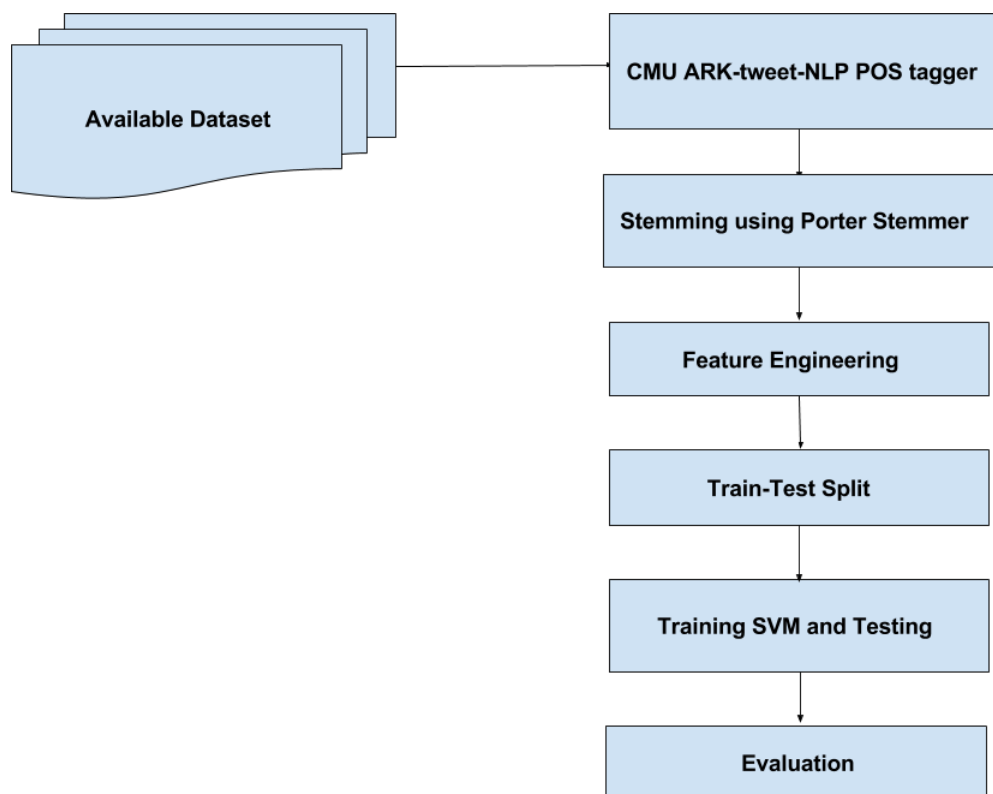


Figure 2.3: Schematic of NLP feature based model

2.3 NLP Feature based Approach

2.3.1 Motivation

Features used in classification task play a huge role in determining the accuracy. Hence one of the most important aspect of classification is feature engineering. Basic tf-idf features usually prove to fall short in terms of finding distinguishing features among classes. Also the larger dimensionality of tf-idf features often pose a problem. Low level lexical features can help in a great way to resolve the issue of tf-idf.

2.3.2 Model details:

The basic work flow of the model can be seen in the sketch of Figure 1.3. CMU ARK-tweet-NLP POS tagger is used on the dataset first to get Part-of-speech of all the words in tweets. Then stemming is done to remove multiple forms of same word which will help in feature engineering process. After getting the feature vectors, the data is splitted into train and test and a SVM with linear kernel is used as classifier. The detail of feature engineering is discussed below.

2.4 Results:

The experimental result of all the non-deep learning methods are shown here for comparison. After analyzing the results, the possible reasons for errors can be summarized as follow:

Table 2.1: Examples of features used

| Sl No. | Feature |
|--------|---|
| f1 | Usage of words like "provide" or "distribute" followed by proper noun(^) or numeral(\$) or common noun(N) |
| f2 | Usage of numerals like contact numbers and helplines |
| f3 | Usage of words like "send" or "need" followed by proper noun(^) or numeral(\$) or common noun(N) |
| f4 | Presence of words like "damage", "destroy", "restore" |
| f5 | Usage of preposition followed by location name |

Table 2.2: Results Table

| Model | Overall Accuracy |
|-----------------|------------------|
| Bootstrapping | 29.48% |
| LDA | 32.11% |
| SLDA | 42.18% |
| NLP-feature | 47.66% |
| NLP-feature+LDA | 48.14% |

1. Many of the miss-classified tweets contain multiple fragments each of which belongs to 2 different classes.
2. The descriptions of few classes [FMT1(re. avail), FMT2(re. reqd.), FMT3(med. re. avail) and FMT4(med. re. reqd.)] are very close thus leading to miss-classification.
3. The number of tweets belonging to FMT4(Medical resource required) in the dataset is exceptionally low and at the same time it has high similarity with FMT2 and FMT3 thus leading to really poor classification accuracy.

Chapter 3

Deep Learning Approach

3.1 Motivation

The process of feature engineering in the task of text classification is time consuming, tedious and often requires a lot of domain knowledge. If the text in consideration is short, finding good features that can distinguish texts into different classes may prove to be difficult. In addition to that, the traditional approach of feature engineering often suffers from the fact that it is very specific to the dataset and does not work well if the dataset is changed. A deep learning based approach to classify tweets posted during disaster events is proposed in this section. We used the same two datasets as before.

3.2 Word-embeddings

Word embedding provides a distributed representation of words in a latent vector space. The embedding vectors are trained on a large corpus and the training process emphasizes on the concept of distributional hypothesis, which indicates that there is a correlation between distributional similarity and meaning similarity of words from a large corpus [13]. Three popular models for word embedding are Googles word2vec, Stanfords GloVe and Facebooks FastText. All the models learn a vector representation of words such that words appearing in similar contexts will have similar representation.

3.2.1 Word2Vec

Word2vec is essentially a predictive model. The word vectors in Word2vec model can be obtained by two means.

- **Continuous Bag of Words(CBOW) method:** In this method, word2vc learns its vectors by minimizing the loss of predicting the target word from the context words given the vector representations.
- **Skipgram method:** In this method, word2vec learns its vectors by maximizing the probability of context words from the centre words given the vector representations. The objective function

can be formulated as:

$$J(\Theta) = -1/T * \sum_{t=1}^T \sum_{j=-m, j \neq 0}^m \log P(w_{t+j} | w_t) \quad (3.1)$$

where Θ represents all the variables to be optimized, m is the window of radius within which the surrounding words are to be predicted. The simplest formulation for $P(w_{t+j} | w_t)$ can be written as follows:

$$P(o|c) = \frac{\exp(u_o^T * v_c)}{\sum_{w=1}^V \exp(u_w^T * v_c)} \quad (3.2)$$

where v_c and u_o are vectors of centre word indices c and context word index o respectively.

3.2.2 GloVe

GloVe learns its vectors by essentially doing dimensionality reduction on the co-occurrence counts matrix. This co-occurrence counts matrix can be of two types namely window-based and full document. Window-based co-occurrence matrix uses window around each word and captures both semantic and syntactic information whereas full document co-occurrence matrix captures general topics thus leading to LSA.

3.2.3 FastText

FastText differs from the above two models in the fact that it takes into account the n-grams while formulating the objective function. This gives FastText the advantage over other models in case of out-of-vocabulary(OOV) words. While Word2Vec and GloVe represent a OOV word as a vector of zeros, FastText can provide a non-zero representation. This is due to the fact that fasttext represents words as sum of their character n-grams. The skipgram training procedure of fasttext is similar to that of word2vec. The probability of a context word given a word is modeled by the following equation

$$P(c|w) = \frac{e^{h_w^T * v_c}}{\sum_{k=1}^K e^{h_w^T * v_k}} \quad (3.3)$$

where feature of a word h_w is computed using both n-gram features and word features. This is where fasttext differs from word2vec. Theoretically, if we only take into account word features in h_w , fasttext will be same as word2vec.

It is observed that word2vec performs slightly better in semantic tasks than fasttext but fasttext performs way better in syntactic tasks than word2vec.

3.3 CNN Model

Convolutional Neural Network is more popularly used in Computer Vision related tasks to find out localized features from images. For NLP, using CNN can be equivalent to finding vectors for every possible phrases. A single layer CNN is composed of a convolution layer and a pooling layer. In convolution layer, multiple feature map can be used. In NLP task, for each of bigram, trigram, 4-gram

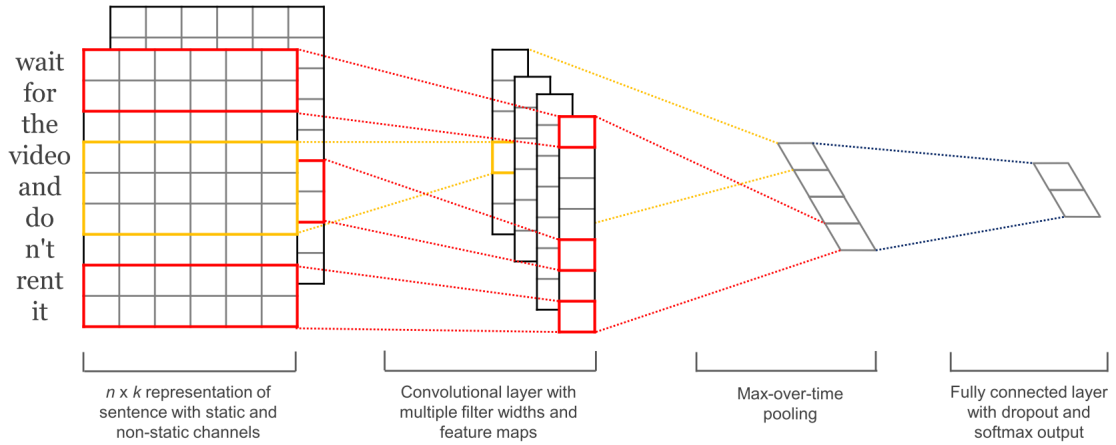


Figure 3.1: The sketch of a CNN network

etc, multiple feature maps can be used to capture different features. Each feature map is supposed to be activated by different characteristic phrases of the text. Max pooling layer can be of many types such as max pooling, min pooling, average pooling, L1 pooling, L2 pooling etc. In Max pooling, the maximum value is chosen from each feature map. Similarly in min pooling, the minimum value is chosen from each feature map and so on. In case of classification task, the flattened max pooling layer can be connected with a fully connected layer to output the probabilities of the text belonging to each class.

3.3.1 Model details:

Each sentence is padded first to make sure they are of same length. Then each word in a sentence is represented as word vector using GloVe embedding. This forms the input layer to the architecture. The input layer is followed by 2 convolution layers with batch normalization. Relu activation function is used in both the convolution layers. Each convolution layer has 32 feature maps for each of 2,3,4,5 and 6-grams. A max-pooling layer is used after second convolution layer. Finally a fully connected layer with 7 neurons is used with sigmoid activation and binary cross-entropy loss to classify tweets into 7 different classes. The model has a total of 830,735 parameters out of which 46,407 are trainable parameters. For the FIRE2017, a slightly modified version of the same network is used resulting in a total of 601,346 parameters out of which 44,418 are trainable.

Both the dataset are splitted into train-validation-test partition the details of which are given in Dataset section. Training is done for 50 epochs with a mini batch size of 64 for both the dataset.

3.3.2 Results:

The class specific results for both FIRE2016 and FIRE2017 are presented below: The table 3.1 shows details of FIRE2016 dataset and the table 3.1 shows details of FIRE2017 dataset.

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .72 | .50 | .59 |
| FMT2 | .71 | .43 | .54 |
| FMT3 | .83 | .38 | .52 |
| FMT4 | 1.00 | .08 | .15 |
| FMT5 | .64 | .27 | .38 |
| FMT6 | .77 | .29 | .42 |
| FMT7 | .89 | .65 | .75 |

Table 3.1: Results of FIRE2016 dataset

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .94 | .94 | .94 |
| FMT2 | .74 | .42 | .53 |

Table 3.2: Results of FIRE2017 dataset

3.4 LSTM Model

When we deal with inputs with no fixed length, the best choice usually is recurrent neural networks(RNN). Different variations of RNN such as gated recurrent unit (GRU), long-short term memory(LSTM) etc have been used successfully in various NLP tasks such as text classification, neural machine translation, text generation etc. RNN scans through the data from left to right and the parameters it uses for each time step are shared across all the time-steps. In a basic RNN, we define two weight matrix, W_{aa} and W_{ax} and bias term b_a . And at each time step $t_i \in t_1, t_2, \dots, t_n$ we update the hidden state as follows:

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (3.4)$$

where x_t is input, a_{t-1} is the hidden state at previous time step. We can write equation 1 in a more compact way by combining W_{aa} and W_{ax} into a single weight matrix W_a such as:

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a) \quad (3.5)$$

where dimension of W_a is (a, (dim(input)+dim(hidden state))). RNN is trained using the method called Backpropagation through time (BPTT). The main problem that arises during the training step of RNN is that the gradient often becomes either too large (exploding gradient) or too small (vanishing gradient) thus making RNN very bad for long term sequences. The detailed exploration of the problem was done by Hochreiter(1991) and Bengio, et al (1994). In practice, LSTM, first introduced by Hochreiter and Schmidhuber in 1997, proves to be superior than basic RNN in modelling long term sequences. LSTM also does hidden state update like RNN but this update is carried

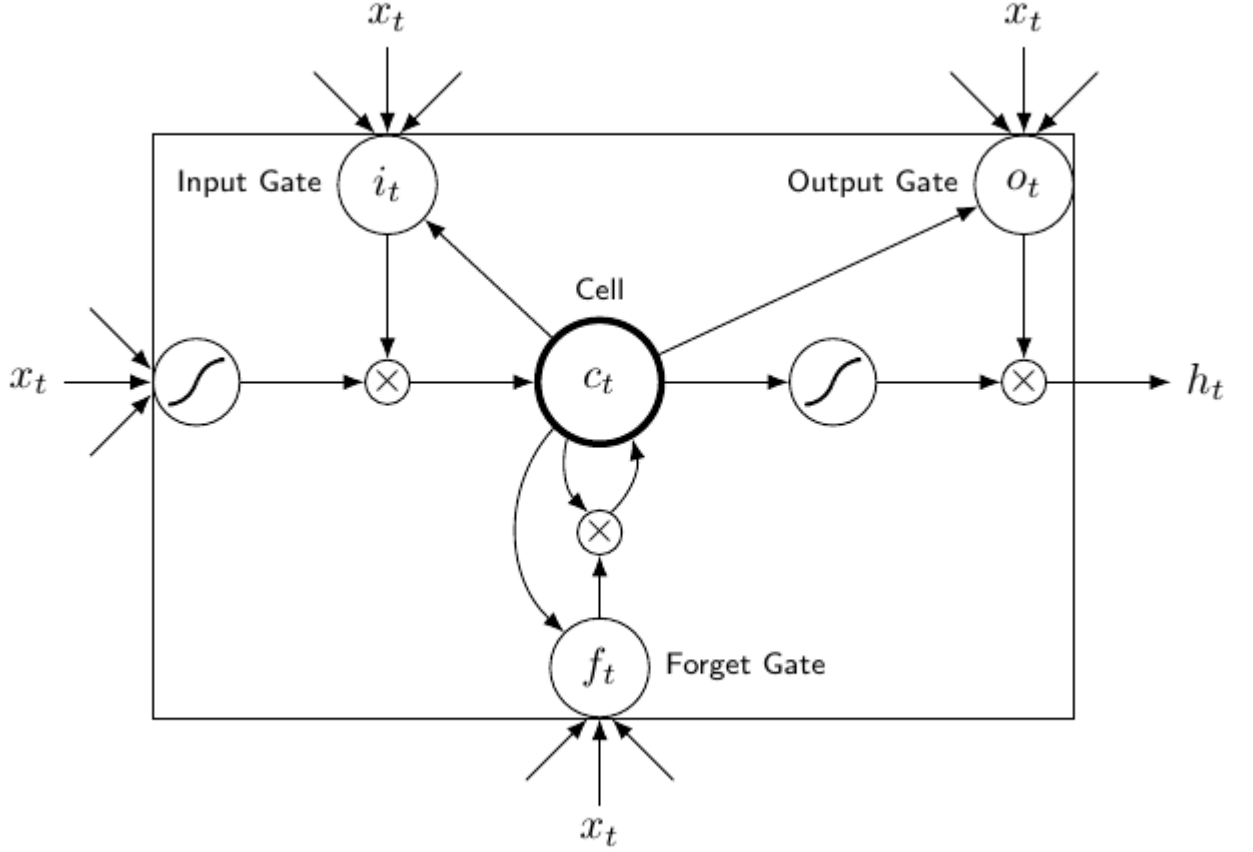


Figure 3.2: Block diagram of a LSTM cell

out in a different way. They are explicitly designed keeping the long-term dependency problem in mind. When we unroll a LSTM through time, it also shows chain-like structure like RNN but the recurring module has a different structure altogether. The core idea behind LSTM is it's 'memory cell' [15]. A LSTM cell is consisted of different gates and a cell-state which let LSTM add or remove information as it seems fit. Information can also pass through unchanged. All this is achieved by the help of three gates namely input gate(Γ_u), forget gate(Γ_f), output gate(Γ_o) and the cell state(C_t).

- **Input gate:** It determines how much information of the current time-step input should be used.
- **Forget gate:** It determines how much information from the previous time step should be retained. Update and forget gate together set the current cell state.
- **Output gate:** Once the current cell state value is calculated, output gate determines how much of the current cell state should be outputted.

The update performed at each timestep $t_i \in t_1, t_2, \dots, t_n$ can be expressed by the following set of equations where W_c , W_u , W_f , W_o , b_c, b_u, b_f and b_o are parameters to be learned.

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \quad (3.6)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \quad (3.7)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (3.8)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \quad (3.9)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \quad (3.10)$$

$$a^{<t>} = \Gamma_0 * \tanh c^{<t>} \quad (3.11)$$

Graves (2012) and Greff et al. (2015) provides a much detailed analysis on LSTM model and it's variants.

3.4.1 Model details:

For FIRE2016 dataset, we use a 5-layer network. Figure 3.3 shows the architecture considered in our work. The first hidden layer consists of Bidirectional LSTM. Bidirectional LSTM basically contains a forward recurrent component and a backward recurrent component which enables it to take into account information from both earlier and later in the sequence before taking a decision, at any point in time. Looking at each sentence from either direction helps Bidirectional LSTM encode textual information better than basic LSTM.

After the first layer of bidirectional LSTM, we have 3 more layers of LSTM each having 256 dimensional hidden states. To reduce overfitting, we use a dropout of .5 in each LSTM layer including the first bidirectional LSTM layer. The final layer is a fully connected layer of 7 neurons to help classifying tweets into 7 classes. We apply sigmoid activation function with binary cross-entropy loss in this layer. The reason behind using sigmoid instead of softmax function which is most popular for multi-class classification problems is that our problem is also a multi-label classification problem. So we treat loss at each neuron in the final layer as independent. We use adam optimizer to train our model for 50 epochs with a mini-batch size of 128. In this model, we have a total of 2,775,815 trainable parameters. FIRE2017 has only 2 different classes and comparatively fewer data-points than FIRE2016. So we use a reduced version of FIRE2016 model which has one less number of hidden LSTM layer thus reducing number of trainable parameters to 2,249,218.

3.4.2 Results:

The class specific results for both FIRE2016 and FIRE2017 are presented below: The table 3.3 shows details of FIRE2016 dataset and the table 3.4 shows details of FIRE2017 dataset.

3.5 GRU Model

Gated Recurrent Unit or GRU is the simplified version of LSTM which is also known to perform well in modelling long sequences. GRU consists of 2 gates namely Update gate(Γ_u) and Reset Gate(Γ_r). Reset gate helps in calculating candidate cell state($\tilde{c}^{<t>}$) at each time step. Update gate determines what part of information from candidate cell state($\tilde{c}^{<t>}$) and previous cell state ($c^{<t-1>}$) are to be used to make current cell state($c^{<t>}$) using Equation 1.15. The point to be noted here is that cell state($c^{<t>}$) is same as the output($a^{<t>}$) unlike LSTM where both of them are different.

The update performed at each timestep $t_i \in t_1, t_2, \dots, t_n$ can be expressed by the following set of

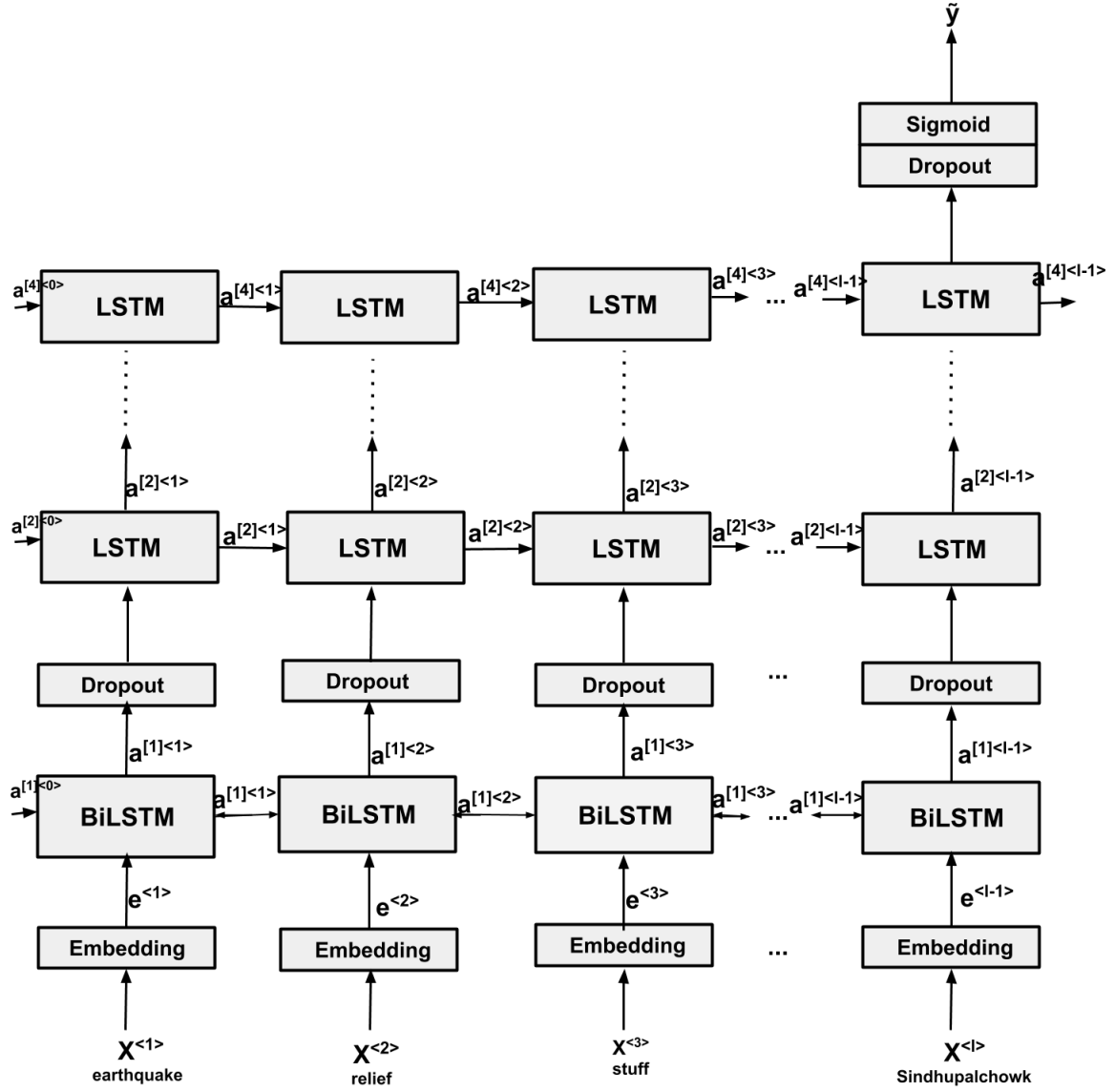


Figure 3.3: Schematic of the architecture used in our work. The figure shows how an input tweet “earthquake relief stuff ...Sindhupalchowk“ is passed to the model (FIRE2016). The embedding layer converts each word into its corresponding GloVe vector representation. This vector is passed to Bidirectional LSTM cell which changes input dimension to latent dimension and passes it to hidden LSTM layer. After the tweet is passed through 2 more hidden LSTM layers, it finally goes through a fully connected layer of seven neurons with sigmoid activation function. These neurons correspond to the seven classes in FIRE2016 dataset. The output of the final layer, \tilde{y} , is a vector of seven dimensions where each element in the vector tells us the probability of the tweet belonging to a particular class.

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .77 | .68 | .73 |
| FMT2 | .85 | .69 | .76 |
| FMT3 | .67 | .86 | .75 |
| FMT4 | .61 | .61 | .61 |
| FMT5 | .70 | .31 | .43 |
| FMT6 | .53 | .64 | .58 |
| FMT7 | .88 | .95 | .91 |

Table 3.3: Results of FIRE2016 dataset

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .96 | .93 | .94 |
| FMT2 | .77 | .84 | .80 |

Table 3.4: Results of FIRE2017 dataset

equations:

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \quad (3.12)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u) \quad (3.13)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r) \quad (3.14)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \quad (3.15)$$

$$a^{<t>} = c^{<t>} \quad (3.16)$$

3.5.1 Model Details:

For this experiment, the network used is largely similar to that of LSTM architecture explained above. After the first layer of bidirectional GRU, 3 more layers of GRU each having 256 dimensional hidden states and dropout regularization are used. The final layer is a fully connected layer with sigmoid activation function and binary cross-entropy loss.

Number of neurons in the final layer and number of hidden layers differ according to datasets. FIRE2016 and FIRE2017 have 7 neurons and 2 neurons in their final layer respectively. FIRE2017 architecture has one less number of hidden layer compared to FIRE2016.

3.5.2 Results:

The class specific results for both FIRE2016 and FIRE2017 are presented below: The table 3.5 shows details of FIRE2016 dataset and the table 3.6 shows details of FIRE2017 dataset.

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .75 | .61 | .67 |
| FMT2 | .78 | .63 | .70 |
| FMT3 | .59 | .81 | .68 |
| FMT4 | .55 | .54 | .54 |
| FMT5 | .66 | .37 | .47 |
| FMT6 | .47 | .59 | .52 |
| FMT7 | .81 | .83 | .81 |

Table 3.5: Results of FIRE2016 dataset

| Class Name | Precision | Recall | F1-Score |
|------------|-----------|--------|----------|
| FMT1 | .95 | .91 | .93 |
| FMT2 | .76 | .82 | .79 |

Table 3.6: Results of FIRE2017 dataset

| Model | Overall Accuracy |
|-------|------------------|
| LSTM | 65.2% |
| GRU | 62.1% |
| CNN | 51.6% |

Table 3.7: Overall Accuracy of Deep Learning models on FIRE2016

Chapter 4

Experimental Details

4.1 Dataset Details:

The dataset used in our experiment is obtained from Forum for Information Retrieval Evaluation 2016 (FIRE2016) and 2017 (FIRE2017). FIRE2016 dataset has approximately 2100 labeled tweets belonging to 7 classes and FIRE2017 dataset has approximately 900 tweets belonging to 2 classes. Example of a class description in TREC format is given below.

- **<num>** Number: FMT6
- **<title>** WHAT WERE THE ACTIVITIES OF VARIOUS NGOs / GOVERNMENT ORGANIZATIONS
- **<desc>** Description: Identify the messages which describe on-ground activities of different NGOs and Government organizations.
- **<narr>** Narrative: A relevant message must contain information about relief-related activities of different NGOs and Government organizations in rescue and relief operation. Messages that contain information about the volunteers visiting different geographical locations would also be relevant. However, messages that do not contain the name of any NGO / Government organization would not be relevant.

Details about the number of tweets are given in the table below.

A few examples of tweets from FIRE2016 from each class are given below to have a clear idea about the data.

4.2 Preprocessing:

We pre-process the raw text before passing it to the embedding layer using the following steps:

1. Remove all the special characters such as '#', '@', '\$' etc
2. Translate nepali and hindi texts to english text. We use google translate api for this task.
3. Remove emoticons and non-ASCII characters
4. Remove URLs

Table 4.1: FIRE2016 Class Details

| Class # | Title | Train | Test |
|---------|---------------------------------------|-------|------|
| 1 | Resources Available | 401 | 175 |
| 2 | Resources Required | 210 | 81 |
| 3 | Medical Resources Available | 231 | 100 |
| 4 | Medical Resources Required | 75 | 36 |
| 5 | Resources Specific Locations | 135 | 53 |
| 6 | Activities of NGOs / Government | 252 | 119 |
| 7 | Infrastructure Damage and Restoration | 178 | 74 |

Table 4.2: FIRE2017 Class Details

| Class # | Title | Train | Test |
|---------|----------------------|-------|------|
| 1 | Need related | 461 | 207 |
| 2 | Availability related | 148 | 55 |

Table 4.3: Examples of tweets from FIRE2016 Dataset

| Class# | Tweet text |
|----------------------------|--|
| Resources Available | Earthquake relief stuff distribution in Baramchi-8,Sindhupalchowk |
| Resources Required | Urgently needed some volunteers who have good command over english language for #earthquake relief contact 9840093843 |
| Medical Resources Required | RT @drspagarwal: Nepal earthquake victims in urgent need of Tarpaulins. Indian #RedCross provides 20,000 more Tarpaulins. #IFRC #ICRC |
| Medical Resource Available | NA medics treating victims of Earthquake in the medical camp established at Piyutar Laitpur . http://t.co/ZXxBUJITSK |
| NGO activity | KISC is setting up a relief fund for our local staff who have been affected by the earthquake, in particular... http://t.co/Ooy1laSX0p |
| Damage and restoration | 116 heritage monuments damaged in April 25 earthquake in Bhaktapur https://t.co/i8NHsCKByi |

5. Convert each word into lowercase.

In order to efficiently use vectorization, we find out the maximum sentence length in the corpus and pad each sentence to have same length.

Chapter 5

Results and Comparison

Various insights that are found from the experiments are discussed in this section. LSTM architecture(Section 1.4) shows the best result out of all the experiments. The result obtained from LSTM architecture is compared with 5 different methods mentioned below.

5.1 Details of State-of-the-Art methods considered for comparison

1. **tf-idf model:** In this model, each tweet is represented by tf-idf vector and SVM with linear kernel is used as classifier.
2. **tf-idf with Normalized Entropy Boosting (NE) [10]:** In this approach, tf-idf score of each term is multiplied by entropy score which essentially helps in giving importance to words which are more specific to a particular class than a word which is equally present over all classes during classification task.
3. **tf-idf with Class-normalized Entropy Boosting (CNE) [10]:** This model improves the low recall value of the previous method by giving an additive boost to the tf-idf score.
4. **Entropy-based Category Coverage Difference (ECCD) [14]:** In this model, to obtain the usefulness of a term for different classes, entropy across all the classes is computed for each term.

5.2 Comparison

5.3 Analysis of Results

Precision, Recall and F-Score metrics are used for evaluating the performances of the methods. The overall comparison according to the F-Score measure is presented in Figure 5.6. It can be clearly seen that LSTM architecture outperforms all other methods in both the datasets. Table 5.1 shows the detailed comparison of all the methods in terms of Precision, Recall and F-score. It can be observed that in FIRE2017 dataset, LSTM outperforms rest of the methods in terms of Precision,

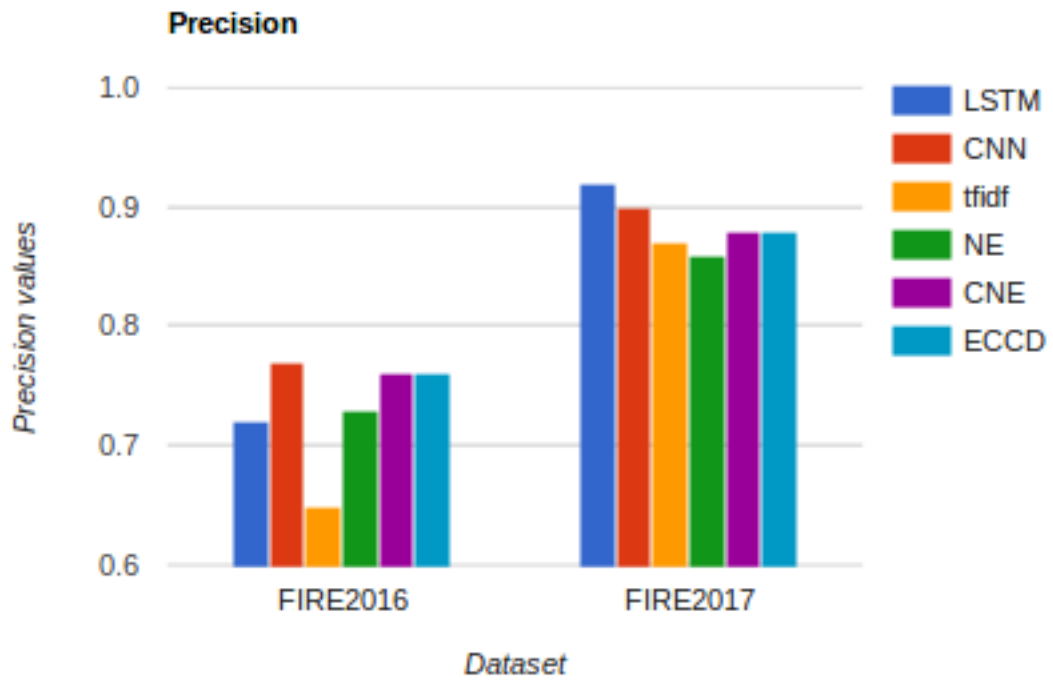


Figure 5.1: Average Precision

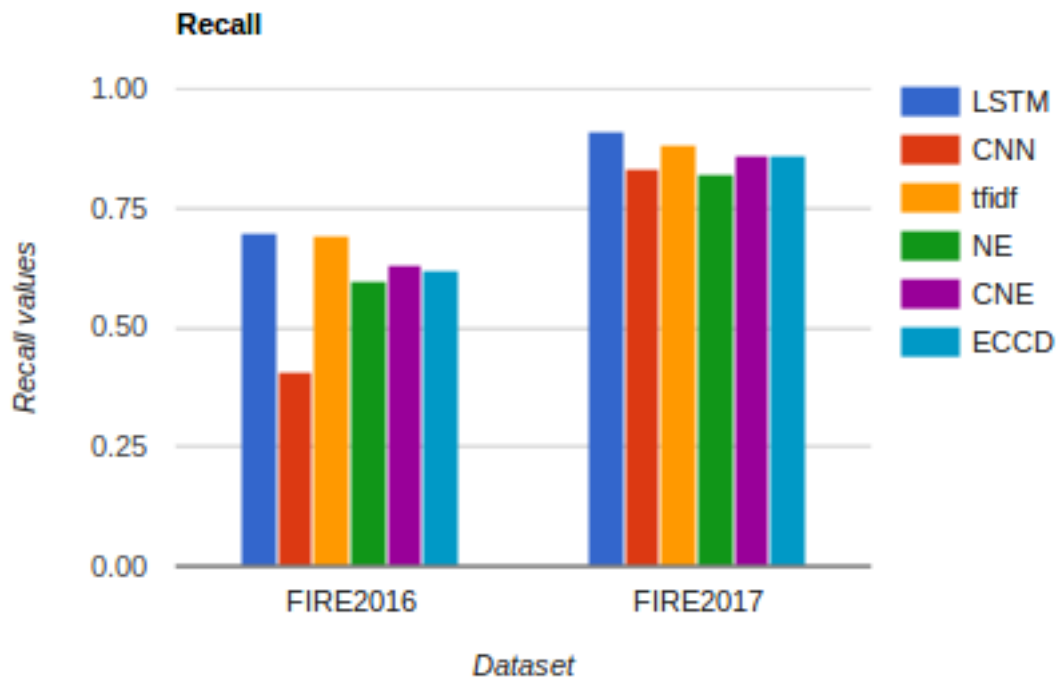


Figure 5.2: Average Recall

Table 5.1: Performance comparison on FIRE2016 and FIRE2017 datasets

| Dataset | Method | Precision | Recall | F-Score |
|-----------------|--------|--------------|--------------|--------------|
| FIRE2016 | LSTM | .7218 | .7011 | .7015 |
| | CNN | .7752 | .4132 | .5210 |
| | tf-idf | .6526 | .6936 | .6701 |
| | NE | .7396 | .6081 | .6535 |
| | CNE | .7643 | .6380 | .6856 |
| | ECCD | .7633 | .6284 | .6801 |
| FIRE2017 | LSTM | .9234 | .9127 | .9159 |
| | CNN | .9028 | .8301 | .8570 |
| | tf-idf | .8703 | .8801 | .8751 |
| | NE | .8647 | .8237 | .8417 |
| | CNE | .8849 | .8692 | .8767 |
| | ECCD | .8831 | .8600 | .8710 |

Recall and F-Score whereas in FIRE2016 dataset, it performs better than the rest in terms of Recall and F-score. CNN performs better than all other methods in FIRE2016 in terms of Precision but at the cost of Recall which ultimately brings down its F-score by quite a margin. On the other hand, the proposed approach performs moderately for Precision, but does way better than others (except tf-idf, which is quite close) according to Recall. As a result, the proposed method has best value of the final F-Score among the methods compared. In Tables 6 and 7, class specific results of Deep LSTM architecture on FIRE2016 and FIRE2017 are shown respectively. It is to be noted that the classes are highly co-related which may account for comparatively low Recall rate for few classes. Another potential cause is that a few classes such as "medical resource required" and "resource required" etc are overlapping in nature.

In terms of overall performance, the LSTM model performed the best. It may possibly be due to the use of the word embedding (which captures semantic information) as the input, and also the automated learning of useful features during the training process using the *appropriate* sequence information. It must be noted that the only use of word embedding as input is not sufficient as CNN fails to perform well in the comparison. Another important point to note from the experiments is that although the data is quite small, the deep learning based methods (specially) achieved good F-Scores. Generally deep learning based models need huge amount of data for training. The use of pre-trained word embeddings was useful for this purpose, as otherwise getting sufficient information about the features could be very difficult for the models.

Using simple RNN shows overall F-Score of 0.53 which is quite worse than Deep LSTM architecture. It is likely due to the fact that simple RNN is not able to model tweets properly.

Using LSTM in the first hidden layer instead of BiLSTM shows slight performance decrease. This shows better modelling power of BiLSTM which incorporates information from both left to right and right to left of a sentence.

Figure 5.5, 5.6 and 5.7 shows the t-SNE plots of the outputs of CNN, LSTM and RNN re-

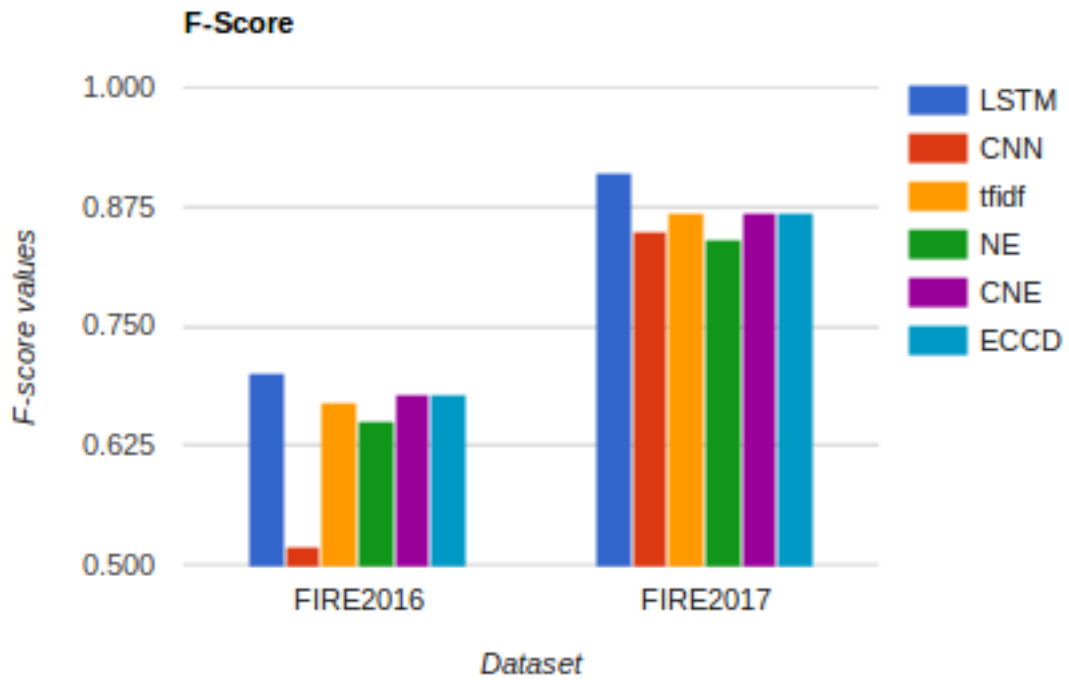


Figure 5.3: Average F-Score

spectively. It can be clearly seen that CNN model is not able to separate different classes properly whereas Deep LSTM model is able to reasonably differentiate between classes which are quite similar such as Medical resources required, Medical resources available.

There are a few tweets in which first part belongs to one class and the second part belongs to some other class. LSTM model is successfully able to classify those tweets into multiple classes whereas traditional models failed to do so. This is due to the efficiency of LSTMs in remembering longer sequences.

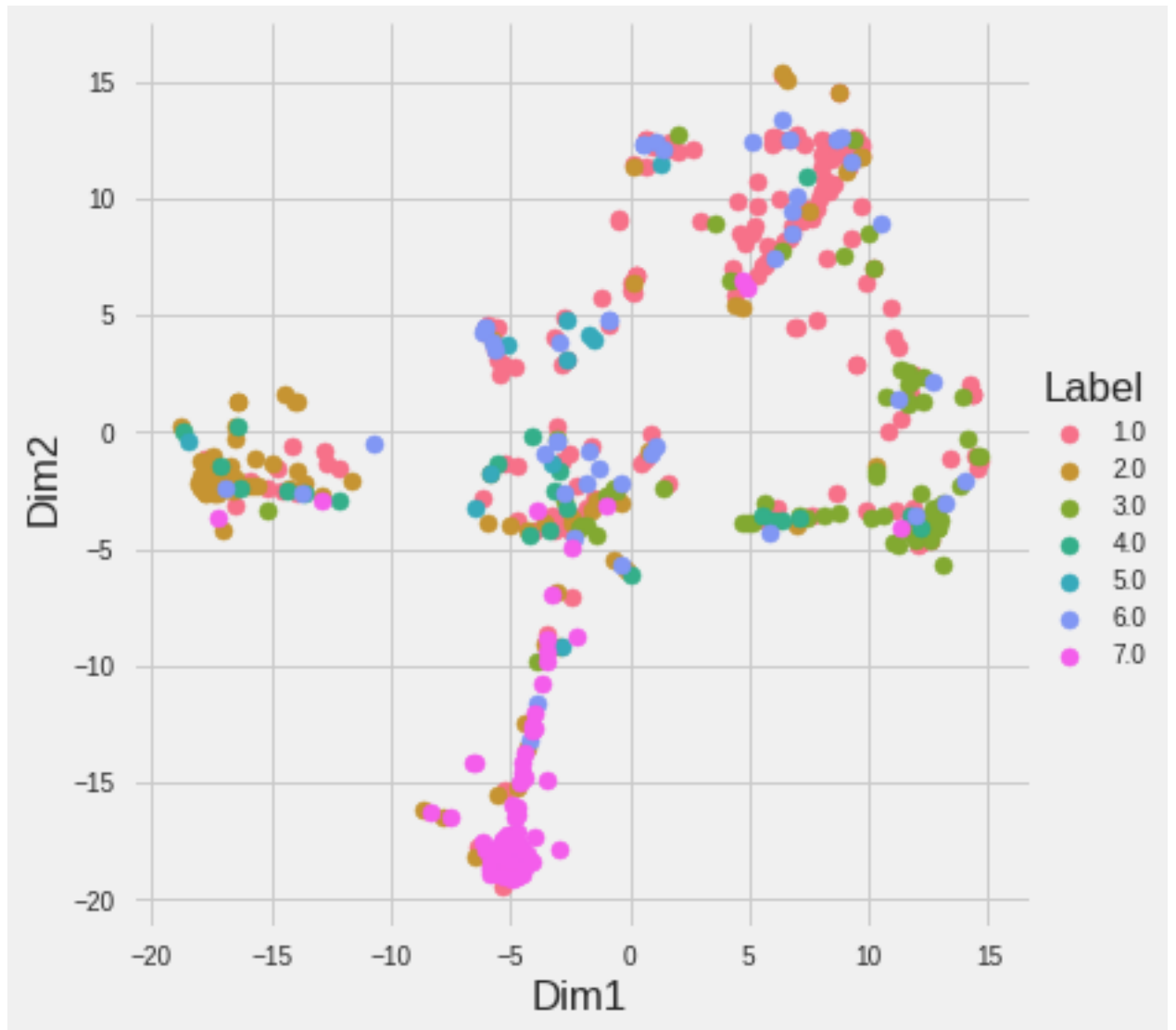


Figure 5.4: t-SNE of CNN

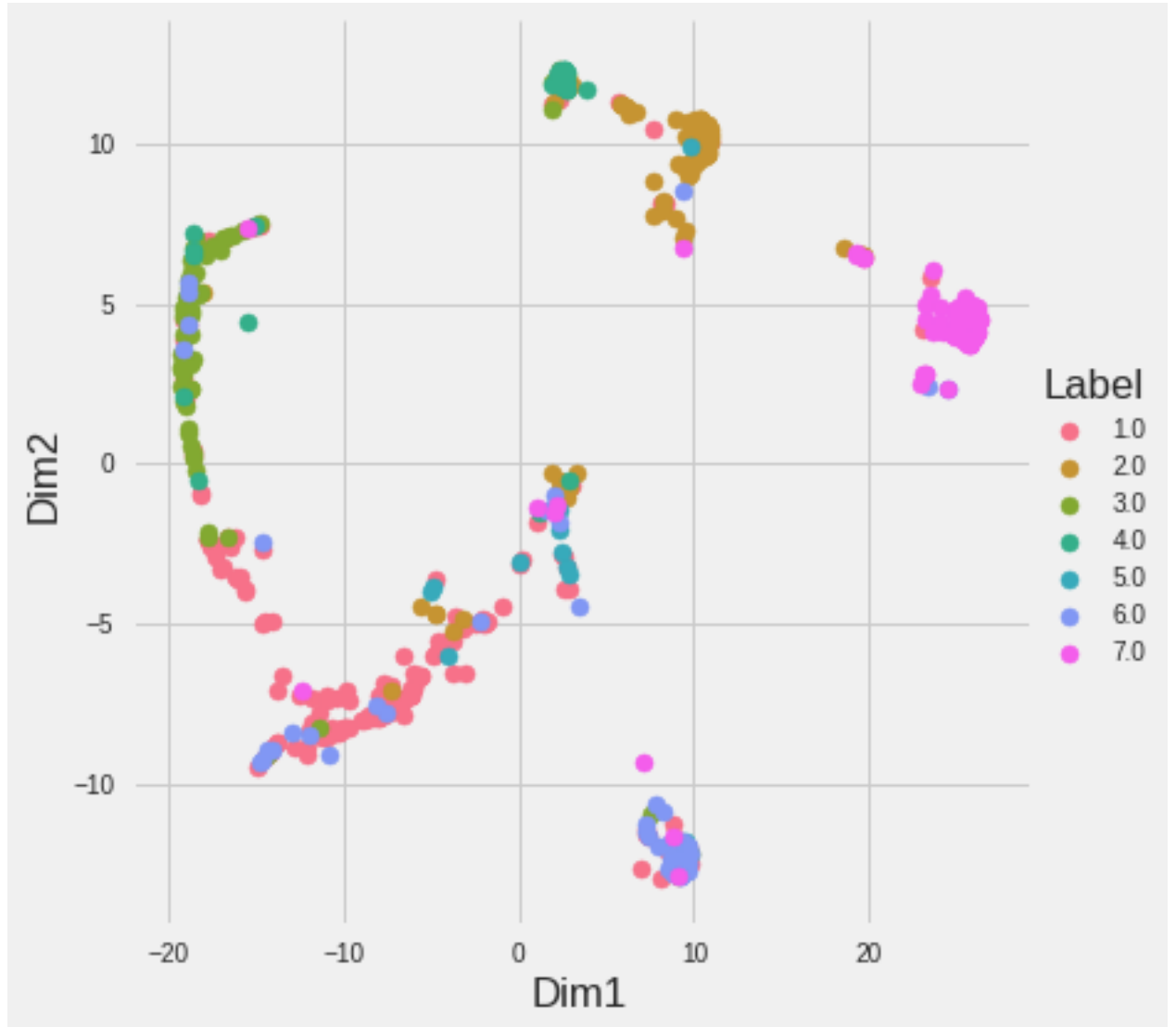


Figure 5.5: t-SNE of LSTM

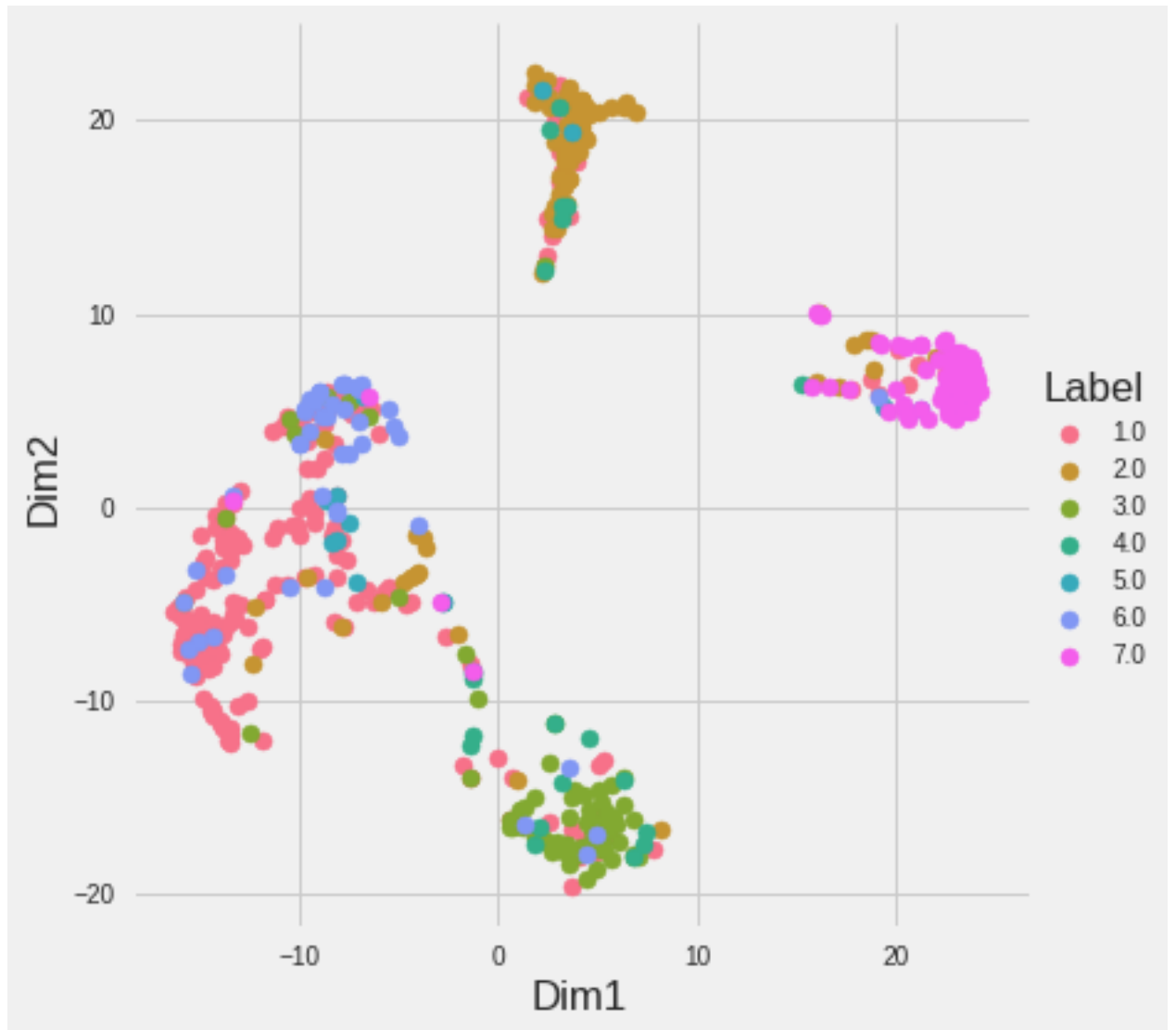


Figure 5.6: t-SNE of RNN

Chapter 6

Conclusion and Future Work

From the experiments, it can be clearly seen that deep learning based LSTM architecture performs better than any non deep learning approaches such as bag-of-words and tf-idf related methods. It is important to note that the proposed architecture is end-to-end and requires minimal modification to use it to a different dataset without any manual feature engineering. Although we observe the model performs reasonably well, it may be interesting to see how we can introduce minor feature engineering along with it to boost performance. Creating different deep learning and non-deep learning based models and use ensemble of them to classify tweets may prove to be useful too. Attention based mechanisms may turn out to be useful too.

References

- [1] Olteanu, A., Castillo, C., Diaz, F., Vieweg, S.: CrisisLex: A Lexicon for Collecting and Filtering micro-blogged Communications in Crises. Association for the Advancement of Artificial Intelligence, Menlo park (2014)
- [2] Singla, R., Modha, S., Majumder, P., Mandalia, C.: Information Extraction from micro-blog for disaster related event. SMERP, ECIR (2017)
- [3] Stowe, K., Paul, M., Palmer, M., Palen, L., Anderson, K.: Identifying and categorizing disaster-related tweets. In: Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media, Austin, TX, November 1 (2016)
- [4] Caragea, C., McNeese, N., Jaiswal, A., Traylor, G., Kim, H.-W., Mitra, P., Wu, D., Tapia, A.H., Giles, L., Jansen, B.J., Yen, J.: Classifying text messages for the Haiti earthquake. In Proceedings of the 8th International ISCRAM Conference, Lisbon, Portugal (2011)
- [5] Fuji Ren and Mohammad Golam Sohrab. 2013. Class-indexing-based term weighting for automatic text classification. *Information Sciences* 236 (2013), 109–125.
- [6] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *Proc. ACM SIGIR*. 959962.
- [7] Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. 2015. Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory. In Proceedings of Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing. 1343–1353.
- [8] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting Rumors from Microblogs with Recurrent Neural Networks. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI16). 3818–3824.
- [9] Prannay Khosla, Moumita Basu, Kripabandhu Ghosh, Saptarshi Ghosh. 2017. Microblog Retrieval for Post-Disaster Relief: Applying and Comparing Neural IR Models. *SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17)*.
- [10] Samujjwal Ghosh, Maunendra Sankar Desarkar. 2018. Class Specific TF-IDF Boosting for Short-text Classification: Application to Short-texts Generated During Disasters. In Proceedings of the The Web Conference 2018. 1629-1637

- [11] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, Sentiment Analysis of Twitter Data, LSM '11 Proceedings of the Workshop on Languages in Social Media, Pages 30-38
- [12] Anirban Sen, Koustav Rudra and Saptarshi Ghosh, Extracting Situational Awareness from Microblogs during Disaster Events
- [13] Sahlgren, M., 2008. The distributional hypothesis. Italian Journal of Disability Studies, 20, pp.33-53.
- [14] Largeton, C., Moulin, C. and Gry, M., 2011, March. Entropy based feature selection for text categorization. In Proceedings of the 2011 ACM Symposium on Applied Computing (pp. 924-928). ACM.
- [15] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutnk, Bas R. Steunebrink, Jrgen Schmidhuber. 2015. LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems. 2222 - 2232