Online Deep Learning for Crisis Response using Tweets

Rajiv Ranjan
Dept. of Computer Science
University at Buffalo
rajivran@buffalo.edu

Abhinav Khare
Dept of Industrial and Systems Eng.
University at Buffalo
abhinavk@buffalo.edu

Samarth Kanabar
Dept. of Computer Science
University at Buffalo
skanabar@buffalo.edu

Abstract

Natural and Man-made disasters are a common occurrence. A lot of activity is registered on social media platforms during disasters in the affected areas and websites such as Twitter can provide useful information for disaster response and management done by humanitarian organizations. In this report, we implement a online Deep Convolutional Neural Network developed by Imran et al.[1] to retrieve information from social media posts about disasters. Their methodology classifies tweets into topical classes. We tested their methodology using CrisisNLP dataset and varied the hyperparameters of their implementation to find the optimal set of them. We vary the number of epochs/patience, minibatch size, dropout ratio and learning rate and present our results.

1 Introduction

Recently, social media is changing the way people communicate not only in daily lives, but also during disasters. There is a surge in usage of social media during an emergency in the affected regions. Many people are willing to share the disaster information through social media. Public uses it to communicate, seek information, raise concerns and express sentiments where as the responders use it to communicate important messages to public.

As a result, there is keen research interest in building tools that employ social media data for disaster management. Numerous applications of social media have emerged in post-disaster recovery efforts after the crisis. In our work, we implement the methodology of Imran et al.[1] to predict the informativeness of social media posts and identify various information types from social media posts. Tweets like the following become the motivation for this kind of work. They are talking about gasoline shortage in their areas during Irma. Information like this can be used by the responders to direct gasoline supplies in the shortage affected areas.

"The shelters are full, there is no gas. Tornados could happen, and storm surge is predicted. So what are people supposed to do? Irma "
"Insane..95 percent of Florida trying to leave at one time. Roads r slammed. No gas. No hotels available. Scared to see my neighborhood after irma"

However, to gather meaningful information from tweets, classification of tweets is required. Automatic classification of small texts is a challenging task. It is unstructured, noisy and gigantic and contains a plethora of information.gathering. And in case of tweets, they are short (only 140 characters) , informal, contain abbreviations and spelling mistakes. This makes interpreting the semantics which humans do easily a very hard problems for machines. As Imran et al.[1] explain "Due to the high variability of the data during a crisis, adapting the model to changes in features and their importance manually is undesirable (and often infeasible). Deep Neural Networks (DNNs) to classify the tweets solve these problems. Distributed representation (as opposed to sparse discrete

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

representation) generalizes well. Since the methodology is online and DNN's are distributed representation, this can be a crucial advantage at the beginning of a new disaster, when there is not enough event-specific labeled data."

In their paper Imran et al.[1] use Deep Neural Network (DNN) to classify social media posts into topical classes. In our work we implement their source code publicly available for crisis computing community for further research at: https://github.com/CrisisNLP/deep-learning-for-big-crisis-data" on the CrisisNLP dataset.

In Section 2, we describe the dataset we used. In Section 3, we describe the details of the implementation of the methodology. In Section 4, we describe the experiments we did and finally in Section 5, we discuss the conclusion.

2 Dataset

We use the CrisisNLP volunteer for implementing the methodology. We combined the volunteer labeled and CrowdFlower (a crowdsourcing platform) to build the training set available in the link http://crisisnlp.qcri.org/lrec2016/lrec2016.html. The training set contains tweets from various disasters in 2014-2015 namely California Earthquake, Chilean Typhoon, Chile earthquake etc. and has aproximately 26k labeled tweets. For online learning data and test set we use the Nepal Earthquake data which has 9k tweets. Each instance of the tweet data contain a tweet id, tweet text and label.

The dataset is labeled into 6 different classes. Affected individuals are reports of deaths, injuries, missing, found, or displaced people, Donations and Volunteering are messages containing donations (food, shelter, services etc.) .Infrastructure and Utilities are volunteering offers Reports of infrastructure and utilities damage. Sympathy and Support are messages of sympathy-emotional support. Irrelevant are messages containing useful information that does not fit in one of the above classes irrelevant or not informative, or not useful for crisis response.

3 Methodology

3.1 Data Prepossessing

All characters are normalised to their lower cased forms, truncate elongations to two characters, every digit is spelled to D, all twitter usernames are spelled "userID" and all URLs " HTTP". All punctuation marks except periods, semi- colons, question and exclamation marks are removed. Further, tweets are tokenized using the CMU TweetNLP tool. Then we divide the data into training data, validation/dev data and test data.

3.2 Architecture

Figure 1 describes the Convolutional Neural Network architecture used in the paper. After the tweets are cleaned and preprocessed they are converted to word embedding vectors. Now, the data is fed into the CNN architecture. The CNN architecture uses a filter/ kernel of size 2 for convolution. We use rectified linear units (ReLU) as the activation functions. The output at this step is called a Feature Map. We apply a max pooling of size 4 on the Feature Map. After that, we combine the data in a single vector, and create a dense Layer. The Dense layer data is fed into a softmax function which gives different distributed probability values for each output class. The highest probability value is considered the classified class of the input data.

3.3 Word Embeddings

The problem associated with text data as far as machine learning/deep learning algorithms implementation and execution is concerned is that, a neural network doesn't work on textual or character data. It needs numbers, so it is essential that the input data which is in text form, collected from twitter, is firstly converted into numeric data. The process of such an operation is called Word Embeddings into vector in AI lingua. Let's see what exactly it is. Every word from an input text data is converted into a number, based on their frequency. A word occurring multiple times receives a lower number and vice versa for low frequency words. After that a fixed vocabulary size is selected for further

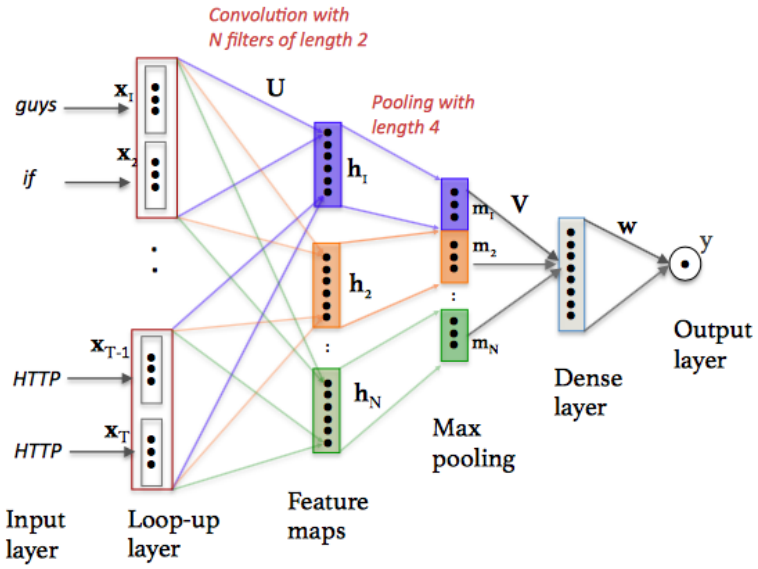

Figure 1: Architecture of the CNN

processing. The paper uses 20 million as it's vocabulary size. After that we convert these numbers into a vector. This vector represents that particular word. The paper uses embeddings vector size of 300-dimensions called the embedded vector.

3.4 Online Learning

Figure 2 describes the online learning algorithm used in the methodology. We first initialize the model parameters θ_0 , which can be a trained model from other disaster events or it can be initialized randomly to start from scratch. As a new batch of labeled tweets $B_t = (s_1...s_n)$ arrives,we first compute the log-loss (cross entropy) in Equation 6 for B_t with respect to the current parameters θ_0 (line 2a). Then, we use back- propagation to compute the gradients $f(_t)$ of the loss with respect to the current parameters (line 2b). Finally, we update the parameters with the learning rate $_t$ and the mean of the gradients (line 2c). We take the mean of the gradients to deal with minibatches of different sizes. Notice that we take only the current minibatch into account to get an updated model. Choosing a proper learning rate "n" can be difficult in practice. Several adaptive methods such as ADADELTA ADAM etc., have been proposed to overcome this issue. In our model, we use ADADELTA.

Algorithm 1: Online learning of CNN

1. Initialize the model parameters θ_0;
2. **for** *a minibatch* $B_t = \{\mathbf{s}_1 \dots \mathbf{s}_n\}$ *at time* t **do**
   - a. Compute the loss $f(\theta_t)$ in Equation 6;
   - b. Compute gradients of the loss $f'(\theta_t)$ using backpropagation;
   - c. Update: $\theta_{t+1} = \theta_t - \eta_t \frac{1}{n} f'(\theta_t)$;

end

Figure 2: Algorithm of online learning

4 Experiment

We use the 26k tweets described in Section 2 as training for the base model and vary the hyperparameters to find an optimal set of them. We vary the number of epochs/patience, minibatch size, dropout ratio and learning rate. For this the data is divided into training/dev/test sets in the ratio 70/10/20.

After having found the optimal set of hyperparameters, for online learning we use the Nepal Earthquake data which has 9k tweets. In the following section we describe our results.

5 Results

Table 1 exhaustively describes the results of our experiment. It shows the hyperparameters varied, hyperparameters kept fixed and the values of precison, recall, f1-score and accuracy. Figure 3 shows the training and validation loss for learning rate = 1, optimiser ="adadelta", patience= 100 and Figure 4 shows it for learning rate = 0.125, optimiser ="adadelta", patience =3. We can see that for a high learning rate equal to 1 the model quickly started overfitting the data with the validation loss increasing and training loss decreasing. From Figure 5-12 show variation in accuracy and F1 score against various hyperparameters. The results are self evident from the graph. We chose the optimal hyperparameters that we found from the experiment and then did online learning on the Nepal Earthquake data. The confusion matrix from our test on Nepal data is shown in Figure 6.

Figure 5: As we increase the mini batch size, accuracy of the model decreases. The most optimum result can be found at a batch size of 135.
Figure 6: As we increase the drop out ratio, accuracy of the model decreases. The most optimum result can be found at a drop out ratio of 0.2.
Figure 7: As we increase the num of epochs, accuracy of the model decreases. The most optimum result can be found at a epoch count of 27.
Figure 8: As we increase learning rate, accuracy of the model decreases. The most optimum result can be found at a learning rate of 0.2.

Figure 9: As we increase the drop out ratio, F1 score of the model first increases then decreases. The most optimum result can be found at a batch size of 0.4.
Figure 10: As we increase the mini batch size, F1 score of the model first decreases then increases. The most optimum result can be found at a mini batch size of 130.
Figure 11: As we increase the num of epochs, F1 score of the model first increases then decreases. The most optimum result can be found at a epoch count of 25.
Figure 8: As we increase learning rate, accuracy of the model increases. The most optimum result can be found at a learning rate of 0.25.

Changing dropout_ratio	**Other hyperparameters**	**precision**	**recall**	**f1-score**	**Accuracy**
dropout_ratio=0.2	minibatch-size=128 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.7087	0.7271	0.694	0.727056586
dropout_ratio=0.4	minibatch-size=128 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.7085	0.7204	0.7021	0.720399429
dropout_ratio=0.5	minibatch-size=128 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.7168	0.7199	0.7009	0.719923918
minibatch-size	**Other hyperparameters**	**precision**	**recall**	**f1-score**	**Accuracy**
minibatch-size=64	dropout_ratio=0.5 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.6964	0.7223	0.7051	0.722301474
minibatch-size=256	dropout_ratio=0.5 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.7052	0.7175	0.7036	0.717546362
minibatch-size=128	dropout_ratio=0.3 filter-nb=150 filt_len=2 pool_len=3 vocab=90 num_of_epochs=25	0.7045	0.7247	0.6963	0.72467903
num_of_epochs	**Other hyperparameters**	**precision**	**recall**	**f1-score**	**Accuracy**
num_of_epochs = 25	dropout_ratio=0.2 minibatch-size=128 filter-nb=150 filt_len=2 pool_len=3 vocab=90	0.7085	0.7299	0.7132	0.729915433
num_of_epochs = 9	dropout_ratio=0.2 minibatch-size=32 filter-nb=150 filt_len=2 pool_len=3 vocab=90	0.7054	0.7273	0.7103	0.727272727
num_of_epochs = 107	dropout_ratio=0.2 minibatch-size=32 filter-nb=150 filt_len=2 pool_len=3 vocab=90	0.6845	0.7014	0.6915	0.701374207

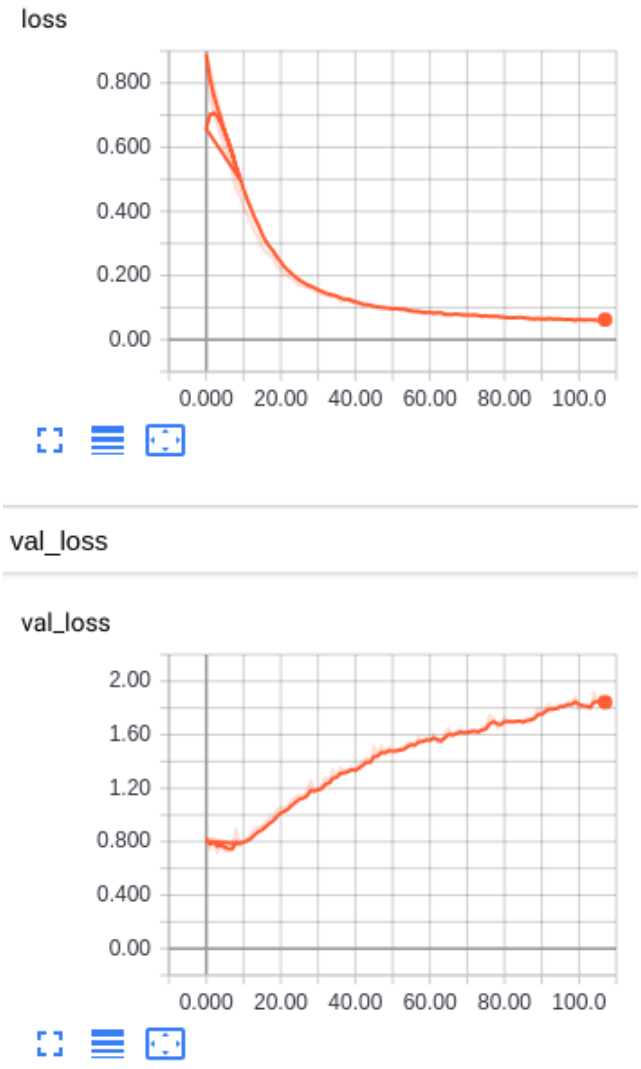

Figure 3: loss vs number of epochs for learning rate =1

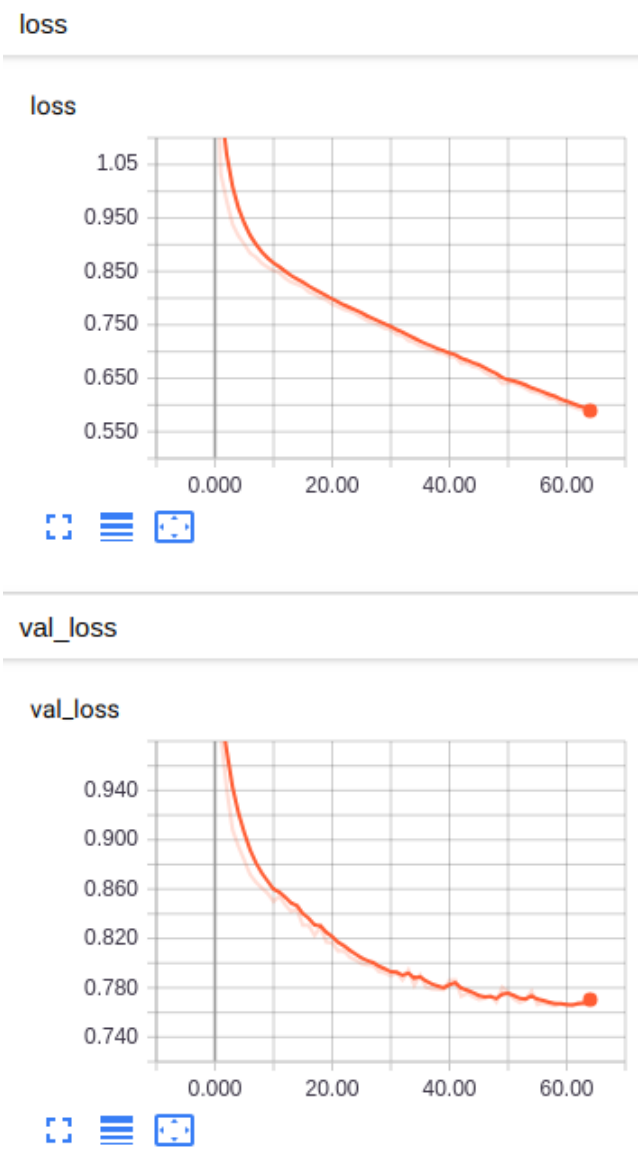

Figure 4: loss vs number of epochs for learning rate =0.125

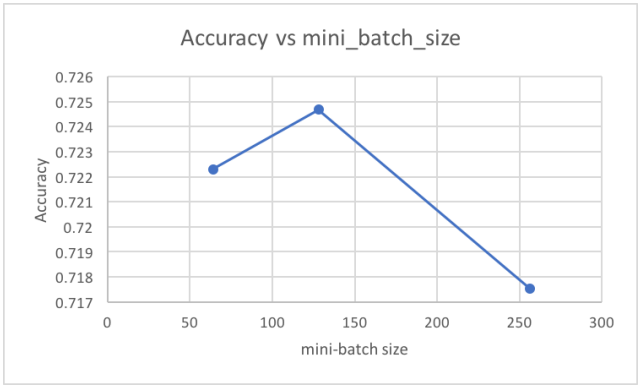

Figure 5: Accuracy vs batch size

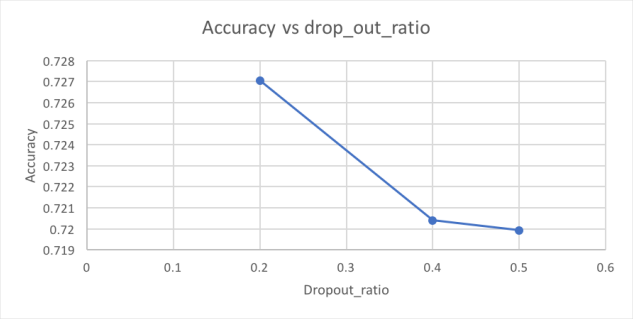

Figure 6: Accuracy vs dropout ratio

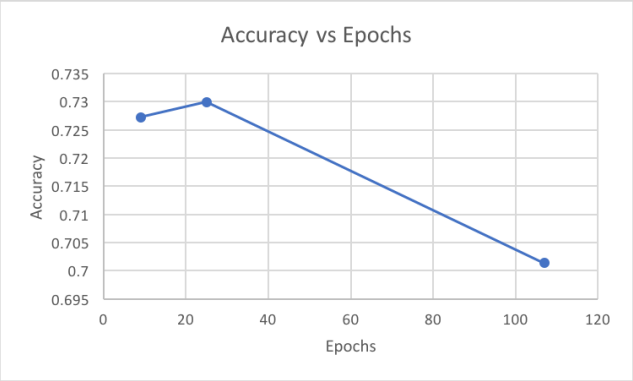

Figure 7: Accuracy vs epochs

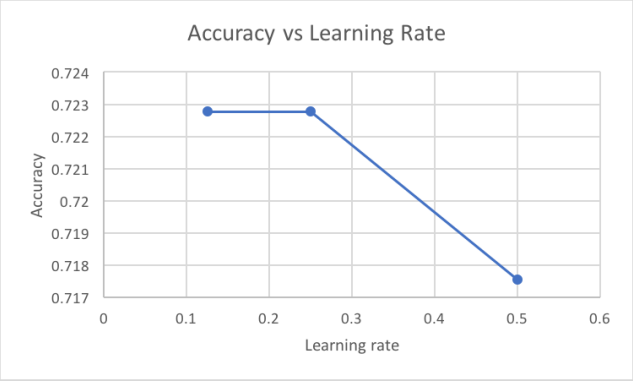

Figure 8: Accuracy vs learning rate

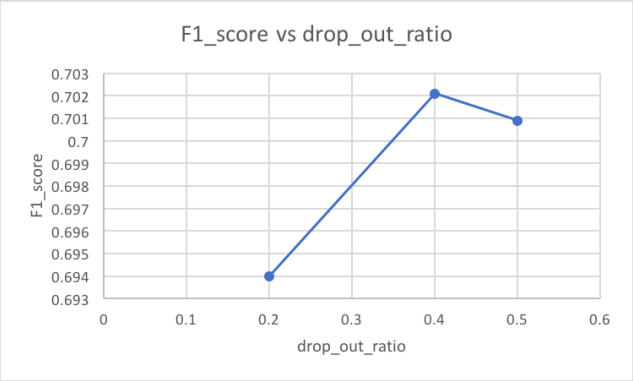

Figure 9: F1 score vs dropout ratio

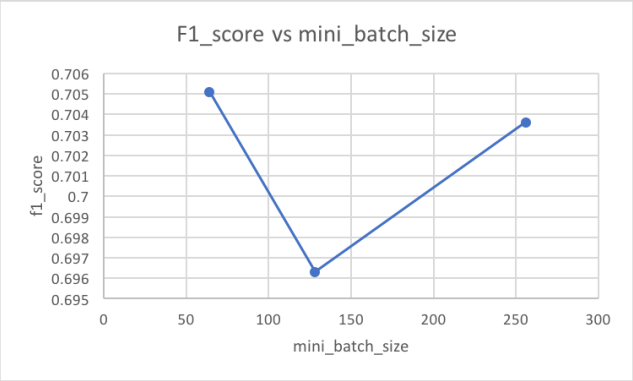

Figure 10: F1 score vs batch size

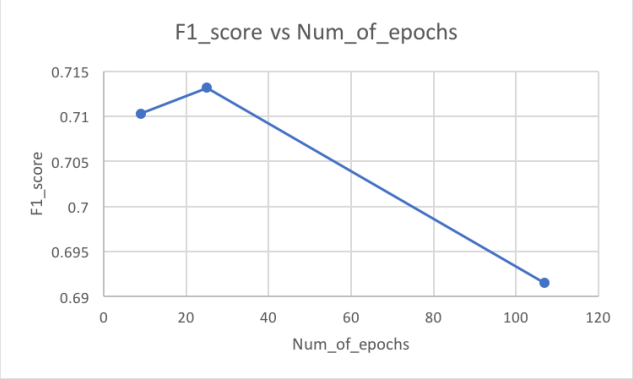

Figure 11: F1 score vs Number of epechs

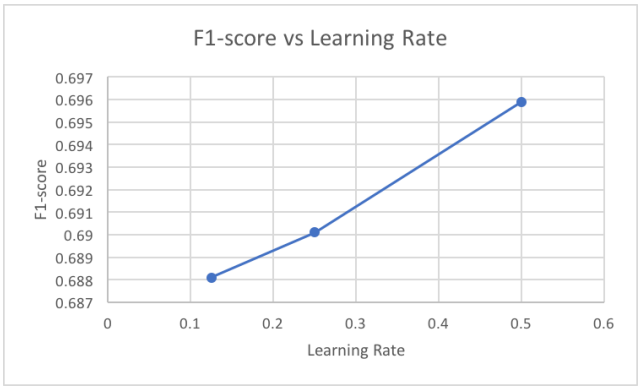

Figure 12: F1 score vs Learning Rate

	affected_individuals	donations_and_volunte	infrastructure_and_ut	not_related_	other_useful_information	sympathy_and_support
affected_individuals	15	0	1	15	17	1
donations_and_volunteering	0	2	1	12	13	0
infrastructure_and_utilities	1	0	15	32	15	1
not_related_or_irrelevant	6	0	4	1237	84	13
other_useful_information	2	3	6	267	221	1
sympathy_and_support	0	0	0	80	4	34

Figure 13: Confusion matrix from test on Nepal Earthquake Data