Classification of Disaster Tweets using NLP

Team: Manasa Hari, Amit Kamboj, Feiyu Cai, Tushar Sharma

**Abstract** — Twitter has been the most important part of digital communication these days. It has been the channel of communication across the world for daily news starting from television gossips to case-of-emergency news. But due to large amounts of data flooding into the twitter base every minute, sometimes, a highly important piece of news such as a disaster occurrence can be submerged amidst not at all important tweets. Due to this, it may take days or even weeks before the disaster news reaches to people, meanwhile, a huge irrevocable loss can happen already. This project aims to highlight the disaster emergency news so that users can be constantly notified on the emergency issues out of millions of other tweets. We take the challenge to build a machine learning model that classifies between tweets about real disasters and the rest. The key challenge is to distinguish metaphorical usage of tragedy vocabulary and the real intended usage of disaster terms. For example, a user tweets ‘Thoughts are a storm, unexpected’. This is clearly a metaphorical statement. Even though it is obvious for humans to interpret that this tweet is not about a real disaster, but it is less clear to a machine. This problem is also an actively ongoing Kaggle competition. We want to explore possible predictors and conclude on the right predictors to solve this classification problem. The dataset has 10,000 tweets that were hand classified[1]. We will explore Naïve Bayes, LSTM and CNN classifiers to solve this problem.

1. Introduction

This report describes the step by step process of building a recurrent neural network (RNN) that classifies tweets to one of two potential classes: Disaster related tweet or Not Disaster related tweet. A basic implementation of a Long Short-Term Memory Network (LSTM) along with a specialized embedding matrix is used in this method. We also explored the variation of test score with input batch size and optimization iterations in this process. We also used a 10-fold cross validation logistic regression with bag of words model which was modified to classify the tweets. This attributes to the inherent ’features’ that the models emphasize.

1. Literature review/related work

In this project, our goal is to make a classification that predicts is the tweet disaster or not. We chose to use neural networks over other classifiers for this project regarding our data input is going to be English words that are way more complicated than just numbers and word of characteristics. Different sentence structures and context can make the same word have different meanings. For this reason, we decided to use the neural network, which is very effective for high dimensionality problems, able to deal with complex relations between variables, non-exhaustive category sets, and complex functions relating input to output variables.

With all the neural network methods, we chose the Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) for this project. First of all, RNN is a generalization of feedforwarding neural network that has an internal memory, which means RNN performs the same function for every input of data with the output of previous outputs. After every computation, the output is copied and sent back into the RNN function internal state (memory). For this reason, this characteristic makes it better for speech recognition while all inputs are related to each other.

A close up of a clock

Description automatically generated

Figure 1: RNN model

However, the internal state cannot last forever. RNN has gradient vanishing while the process takes too long. Gradients are values used to update neural network weights, which will shrink during back-propagation. When the gradient becomes too small, it doesn't contribute to the learning process. So, we use LSTM to resolve this problem. LSTM is a modified version of RNN, which improved in remembering past data in memory. LSTM uses back-propagation to train the model within the LSTM's cell which is better when classify, process, and predict time series given time lags of unknown duration. In each LSTM cell model, 3 gates presented to help to decide what information to keep or throw away:

1. Forget gate: used to decide what detail can be discarded from the block
2. Input gate: used to decide which input should be used to modify the memory
3. Output gate: used to calculate the cell state

A picture containing clock

Description automatically generated

Figure 2: LSTM gates model

1. Project Architecture

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Here, the given datasets are 1) train.csv and 2) test.csv files.

Train.csv:-

A screenshot of a social media post

Description automatically generated

1. Testing methodology/Accuracy testing (5 pts)

<We will add prediction/results and the accuracy from the python notebook>

1. Result, conclusion, future work detailed description (10 pts)

<Results of the Kaggle competition and future work to use it for classification problems>

CMPE-257 Project Developed in Spring 2020

—————————— ◆ ——————————

**References**

1. Data and Problem source: <https://www.kaggle.com/c/nlp-getting-started/data>.
2. Mittal, Aditi. “Understanding RNN and LSTM”. Oct 12, 2019. <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>
3. Phi, Michael. “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation”. Sep 24, 2018. <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

**Dataset and Description**

Link: <https://www.kaggle.com/c/nlp-getting-started/data>

1. Data format: Each sample in the train and test set has the following information:
2. The text of a tweet
3. A keyword from that tweet (although this may be blank!)
4. The location the tweet was sent from (may also be blank)
5. Files we have:
6. train.csv - the training set
7. test.csv - the test set
8. sample\_submission.csv - a sample submission file in the correct format
9. Columns we use:
10. id - a unique identifier for each tweet
11. text - the text of the tweet
12. location - the location the tweet was sent from (may be blank)
13. keyword - a particular keyword from the tweet (may be blank)
14. target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)
15. Output: We will predict whether a given tweet is about a real disaster or not. If so, predict a 1. If not, predict a 0.

**Technologies we plan to use:** Python, Keras, Tensorflow.