|  |
| --- |
| **Word Embeddings and Recurrent Neural Networks**  **For Disaster Tweet Classification** |
|  |
| **Boyang Wei, Lechuan Qiu** |
|  |
|  |
|  |
|  |

Abstract

|  |
| --- |
| 000  001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049 |

Tweet has become an important communication channel in times of emergency. The prevalence of smartphones enables people to announce an emergency when observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter and take actions in time when emergency occurs. Word embeddings and deep neural networks are proven to be useful when training the Tweet texts, after necessary steps of cleaning. We proposed a systematic way for cleaning, training, pruning and evaluating the results from four different methods from state of the art. Specifically, baseline models TFIDF, Word2Vec with Logistic Regression are used to compare to recurrent neural networks LSTM and BERT. The experiment results on Tweet disaster dataset prove that…

Introduction

Tweet served as one of the most popular communication channels for sharing daily events. The format of Tweet and ways it is written could be learned by human and computer to programmatically detect the underlying information. To learn how to extract the information from the raw texts, this paper investigates two traditional approaches and two deep learning methods for text classification, specifically aiming for detect if a Tweet refers to a real disaster or irrelevant information. We also purposed plans for data cleaning specific for Tweet and through initial exploratory data analysis. The results will be evaluated based on common metrics: precision, recall, F-score and support.

Data

Tweet disaster data from Kaggle Competition are used for the entire pipeline from data cleaning, modeling to evaluation. The data consisted of a training and testing file in form of common-separated values. The training and testing data contained four features and one predictor: ID, text, location, keyword and target. The former four features are used to predict the target as label of disaster or non-disaster for the Tweet. Training and testing files contain 3243 and 7503 unique rows.

For four features, ID stands for a unique identifier for each tweet; text is the raw body of Tweet; location is the location the tweet was sent from; keyword is a keyword from the tweet. Through exploratory data analysis, due to large portion of missing values, only text is used for disaster prediction.

Background

Traditional natural language processing for classification problems rely on proper cleaning, careful choice of word embedding and machine learning models. Texts data are treated differently from numerical data which could be statistically normalized. For tweets, texts needed to be ‘normalized’ as proper English and Non-Roman characters sometimes need to be removed. Stop words, symbols and punctuations are usually removed for training. After cleaning the data, this paper examines different combinations of word embeddings and machine learning models. We also proposed some potential explanations on the results from best to worst combinations which provide valuable insights for further Tweet analysis.

Methodology

We first performed a comprehensive exploratory data analysis from the original dataset, which provided guidance on what aspects should we focus on the cleaning part. This section will cover all the operations and explanations behind each step. Section 4.1 will cover all the data cleaning and wrangling methods and 4.2 to 4.5 will introduce word embedding and machine learning algorithms chose for this binary classification problem.

* 1. Data Cleaning

Through initial exploratory data analysis, we found that two classes are not evenly distributed: there are 3271 disaster data and 4342 non-disaster data. Unbalance is resolved by collecting equal number of sample data from both classes. From the EDA part above, we can see most of the key words and locations are missing from training set. Therefore, only texts are used for further classification task. The data cleaning part will focus on using regular expression to remove irrelevant information such as links, HTML tags, symbols, punctuations, stop words. N-grams are also performed in the raw tweet texts. Most of the frequent unigrams, bigrams and trigrams make sense to both disaster and non-disaster classes.

Links are removed since tweet contains hyperlinks that are reflected as irrelevant information or meaningless symbols from main body of texts. Removing HTML tags also help get rid of irrelevant tags that are used for web formatting. Punctuation and stop words are removed as they appear frequently and do not have impacts on the meaning of main body. Emojis, symbols and flags are finally removed since emojis are represented as "\U0001F600-\U0001F64F", which does not reflect the direct emotions which it supposed to function.

Further cleaning including lemmatization and stemming are also performed for standard regular expression cleaning.

* 1. TF-IDF + Logistic Regression

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. In our case, we measure the term frequency for disaster class and non-disaster class and the terms appear frequently tend to have higher values in terms of weight. In general, this word embedding measures the relevance of each word appeared in both classes, and highly relevant words, together, imply the high chance that this tweet belongs to the given class. The general formula for TF-IDF is shown in figure 1.1.





Figure 1.1 TF-IDF Vector Calculation Formula

In exploratory data analysis section, when displaying the top 30 most frequent unigrams, bigrams and trigrams, the frequent words appeared in both classes are terms such as ‘fire’, ‘suicide bomber’ and irrelevant terms such as ‘video’ and ‘cross body’. As these terms make sense to natural language and human, TF-IDF is our first baseline word embedding to use and followed by a relatively simple machine learning model, logistic regression. The result metric from randomly splitting training data is displayed below (Table 1.1)

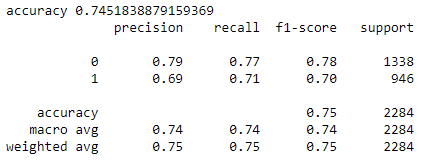


Table 1.1 Classification Report from TF-IDF and Logistic Regression

* 1. W2V + Logistic Regression

Word2Vec is used as our second word embedding for disaster dataset since it is a popular word embedding technique that uses a two-layer neural network. Particularly, the mathematical operations on original texts detect the similarities in-between and cluster words with similar meaning together in vector space. As users write tweet in different ways and the word-choice is diverse among all the users, clustering words with similar meaning reduces the complexity of large training corpus and improve the performance accordingly. For binary classification problem, we used bag of words (CBOW) for training Word2Vec. And the library used is Genism. As for comparing to the baseline model with TF-IDF word embedding, we used default hyperparameters and same training iterations, training and testing split as to the previous model. Logistic regression with same parameters is used for explicitly comparing the performance of word embeddings. The result metric for Word2Vec is displayed below (Table 1.2)

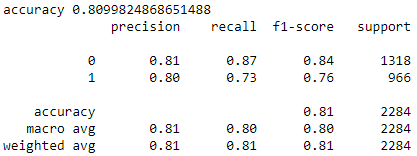


Table 1.2 Classification Report from Word2Vec and Logistic Regression

Long short-term memory (LSTM)

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

Traditional natural language processing pipeline rely on choosing relevant word embeddings and machine learning algorithms and largely depend on the data. Using deep learning frameworks enables more flexible models to detect subtle relationships between body of texts and its target, in this case, the class. We first built customized Long short-term memory (LSTM) layer in a sequential recurrent neural network to compare with the baseline model from traditional approaches.

LSTM is an artificial recurrent neural network architecture that processes and learn the entire sequence of text. Specifically, it carries the local information from previous words as it iterates through the entire sequence, making some important correlations from two ends to be obvious. It also helps with vanishing gradient problem as training traditional RNNs always occurs. In our case, through initial exploratory data analysis part, most of the tweets from both classes are consisted of roughly more than 170 characters. The long sequence of words could have inter-correlation and may be usefully for training purpose. We defined the max number of words to be 30,000 and max number of words in tweet to be 1,000. The word embedding used is W2V as default embedding layer in Python library Keras. The optimizer chose here is Adams since it is so far the best optimizer from all our attempt. The overall customized architecture is shown below in figure 1.2.

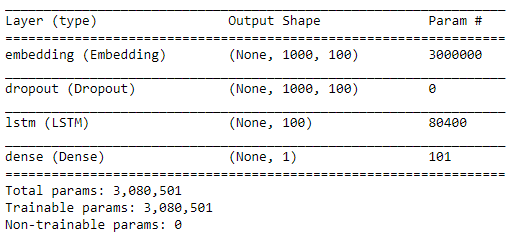


Figure 1.2 Model Summary for customized LSTM Recurrent Neural Network

By training on the same split as in the baseline models, the prediction accuracy and loss began to overfit after epoch number 3, suggesting that the model is over-flexible for our dataset. (Figure 1.3, 1.4)

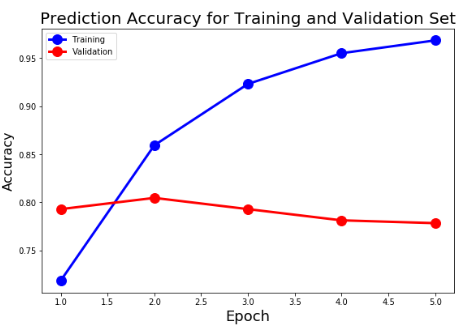


Figure 1.3 Prediction Accuracy for customized LSTM Recurrent Neural Network

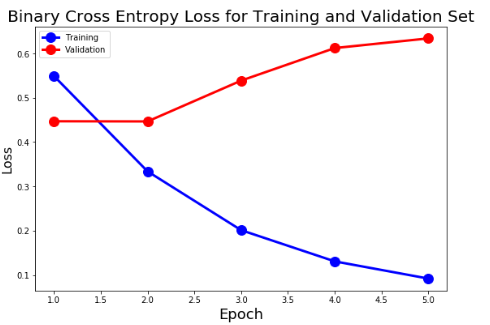


Figure 1.4 Binary Cross Entropy Loss for customized LSTM Recurrent Neural Network

The final validation accuracy is roughly 0.79, which is slightly lower than that of Word2Vec embedding but better than the baseline model of TF-IDF.

Bidirectional Encoder Representations from Transformers (BERT)

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

The second recurrent neural network we used is a state-of-art model from Google AI. The pre-trained model can be adapted for a wide variety of tasks, and binary classification is one of them. BERT’s key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. By training the models with reading the body of texts from forward and backward directions, we believe the model gain further insights and therefore make better predictions on the binary classification task. The pretrained BERT architecture is shown below in Figure 1.5

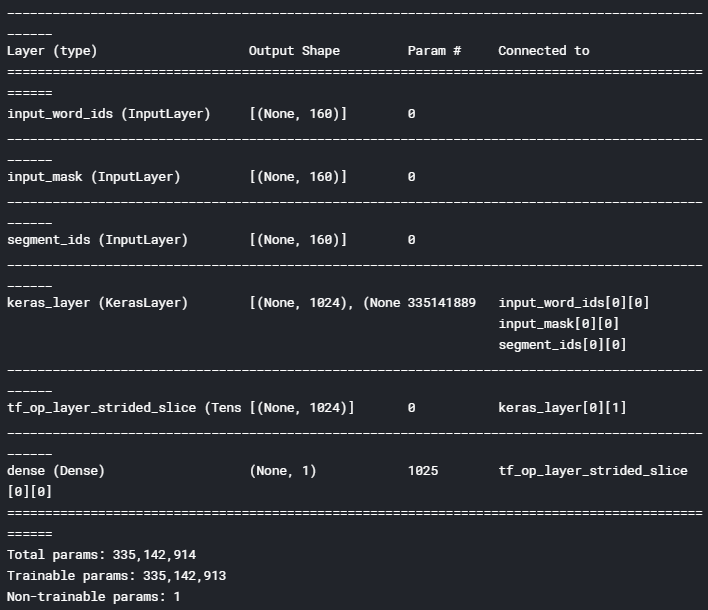


Figure 1.5 Model Summary for Bidirectional Encoder Representations from Transformers

By training on the same split as in the baseline models, the number of epochs chose is 3 based on previous RNN result. The validation accuracy is roughly 0.83, which is the best among all models.

1. Discussion

Based on prediction results and training, validation loss and precision, BERT performed the best among all four models and TF-IDF tends to perform the worst as baseline model.

For TF-IDF, we believe that since tweets are written by multiple users, the common words in each tweet could be a good reference and implication of the class, the overall sample size is still low compared to the total number of tweets. Therefore, TF-IDF potentially captured the most common words and assign the weights but neglect the large portion of terms that also have important inference.

Word2Vec tends to perform better than TF-IDF, given that tweets could be diverse in terms of word choice. Clustering different terms of the similar meaning enable the computer to train relatively simple model. Given that both deep neural networks show overfitting with even regular number of epochs, Word2Vec is a good choice in our sample dataset.

For LSTM and BERT, we believe that this task for classifying disaster and non-disaster tweet is relatively simple. As training log shows the model gets overfit to epoch 3 and 4, the models are clearly too flexible for the tweet. Therefore, deep neural networks, even though BERT scores the highest in term of prediction accuracy, are not necessary and appropriate to use for this task. Simple model such as Word2Vec and logistic regression also scores 0.81 accuracy rate and are easier to interpret. We will continue to work on different word embeddings and deep learning models to better decide if we really need to have complex models to better capture details that traditional models could not learn and interpret.

References

Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276.

Graves, A.; Liwicki, M.; Fernandez, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 31 (5): 855–868. CiteSeerX 10.1.1.139.4502.

Nayak, P. (2019, October 25). Understanding searches better than ever before. Retrieved from <https://www.blog.google/products/search/search-language-understanding-bert/>

Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2.

Manning, C.D.; Raghavan, P.; Schutze, H. (2008). "Scoring, term weighting, and the vector space model" (PDF). Introduction to Information Retrieval. p. 100. doi:10.1017/CBO9780511809071.007. ISBN 978-0-511-80907-1.

"TFIDF statistics | SAX-VSM".