Classification of Tweets Related to Disasters

Project 3: Report Philip Ayazi, Daniel Viassolo

Problem Statement

Our goal is to classify whether tweets are actually related to a disaster or not.

Data-sets

The data was obtained through a Kaggle competition and consisted of roughly 10,000 data points. The data consisted of 4 columns: id, keywords, location and Tweeter text.
The Kaggle link: https://www.kaggle.com/c/nlp-getting-started

Data Wrangling

We downloaded the data set from Kaggle and read it into a pandas dataframe. From here we used gensim preprocessing to clean the data. This was done by making all letters lowercase, removing white spaces, stripping punctuations, and removing stop words. From here we organize all of our cleaned tweets into a 2D array so that our BOW, TD-IDF, and embeddings could analyze the data.

Exploratory Data Analysis

Exploratory data analysis was performed to check the balance between classifications of tweets related to (True) and not related to disasters (False). We found that True and False tweet sets were fairly balanced - 43% and 57%, respectively.

Modeling

Modeling consists of 2 phases: (A) generate **features**; i.e., convert cleaned tweets (lists of words) into numbers (one vector per tweet); (B) use **classifiers** to map these features into a decision on whether any given tweet is True or False.

For phase A, 4 different approaches were used. Two directly based on Bag of Words - Count Vectorizer and TF-IDF, both giving high-dimensionality vectors of more than 10,000 components. Two popular embeddings - Word2Vec and GloVe, generating vectors of dimension 300.

For phase B, 4 different Classifier algorithms were used - Logistic Regression, Random Forest, XGBoost, and a Deep Neural Network. These approaches were selected since they are well-known algorithms, widely used in many applications examples today.

Results for a total of 13 different models are given in the following sections.

Train-Test Split

The data set already had a training set and a test set but because the test set did not have labels of each tweet, we had to further split the training set into a *practical* train set and test set. The data was split 70/30 train/test, using sklearn test train split function.

Results

The following 2 tables summarize our results for each one of the 13 models studied in terms of classification accuracy, f1-score, precision and recall rates. Notice that these tables display the best results - from the point of view of Accuracy and F1 score. Hyperparameter tuning was tried for all Classifiers. For logistic regression, the tuning parameters were the solver, the uniformity, and the l1 ratio. For the random forest we tuned over the number of estimators, maximum features, maximum depth of each tree, and the minimum sample split. For XGBoost, we tuned over number of estimators and learning rate. For the Deep Neural Net, *Hyperas* (a wrapper around hyperopt for fast prototyping with keras models) was applied. For Logistic Regression, Random Forest, and XGBoost, the sklearn functions *GridSearchCV* and *RandomizedSearchCV* were applied.

Accuracy	Logistic Regression	Random Forest	XGBoost	Deep NNet
CountVectorizer	80%	79%	79%	X
TF-IDF	80%	79%	77%	X
Word2Vect	79%	80%	82%	79%
GloVe	80%	82%	82%	X

F1 score	Logistic Regression	Random Forest	XGBoost	Deep NNet
CountVectorizer	75%	71%	74%	X
TF-IDF	75%	72%	69%	X
Word2Vect	75%	74%	78%	79%
GloVe	76%	76%	78%	X

PRECISION RATE %	Logistic Regression	Random Forest	XGBoost	Deep NNet
CountVectorizer	81%	82%	79%	X
TF-IDF	79%	82%	80%	X
Word2Vect	77%	84%	82%	79%
GloVe	78%	86%	82%	X

RECALL RATE %	Logistic Regression	Random Forest	XGBoost	Deep NNet
CountVectorizer	69%	82%	69%	X
TF-IDF	71%	82%	60%	X
Word2Vect	73%	84%	75%	79%
GloVe	73%	86%	75%	X

Conclusions

Looking at accuracy and f1-score, the 2 best models are obtained by using XGBoost classifier with either one of the embeddings - Word2Vec and Glove. In both cases, accuracy = 82% and f1-score = 82%. If we had to decide on one of them, for our application we would look into recall rate (since we prefer to avoid missing true disaster tweets at the expense of a few more false alarms). Looking at recall rates, in both cases we have recall rate = 75%. Another very good option is Random Forest with GloVe embedding, strong from view point of recall and precision rates.

Further Exploration

More work we can do for this project is remove out of vocabulary words which could cause noise in our models. Also, we performed our analysis on the average word vector for each tweet. Going back and creating vectors for each word in each tweet should give us better accuracy for our embeddings.

References

https://towardsdatascience.com/sentiment-analysis-with-text-mining-13dd2b33de27

Sklearn documentation
Gensim documentation
Spacy documention
Class lecutres