Would any intervention authority have an interest in analyzing tweets to detect disaster as soon as possible ?
Machine Learning for Natural Language Processing 2020

BOUNOUADAR Oumaima
ENSAE Student
oumaima.bounouadar@ensae.fr

DUFOUR Camille
ENSAE Studient
camille.dufour@ensae.fr

Abstract

In this project we practiced various NLP techniques and applied machine learning to identify disaster-related tweets.

1 Problem Framing

Our problem is to detect which tweets are about a disastrous event to help to exclusively notify law enforcement officials about urgent emergencies while ignoring reviews of joke, movie review or something non-disastrous.

2 Experiments Protocol

2.1 Cleaning our data

The goal is to decrease computational costs by removing unmeaningful tokens. We face a trade-off between parsimony and relevance. Indeed in the case of tweets, and in the particular case of disaster prediction, capital letters, emotijs punctuations can well be meaningful. We opted for two approaches:
cleaning step by step : we check at every step the effects and the performance of our function. Based on the results of the current step we decide what will be the next step and so on until getting our tweets with just meaningful text.
cleaning all steps in one function : we do not check each step we evaluate only the final result and if it contains unmeaningful tokens we change the order of steps add or delete some steps until we get the desired result.
Cleaning will also be a useful tool to delete "quasi-duplicates". Indeed tweets' database are full of retweets, only different by an url or an hashtag reference.

2.2 Creating training, dev and test sets

We will split the dataset into 3 sets:
Training set: 60% used to train classifiers and obtain a model.
Dev set: 20% used to apply the model and check performance while turning it.
Test set: 20% used as the final performance check to see if the generalization is satisfying.

3 Data modelling

3.1 Bag of words

3.1.1 Countvectorizer or TFiDFvectorizer

With these bag of words embeddings, words are vectorized regardless of their position in the tweets. Countvectorizer uses occurrence counts in a sparse matrix. TF-IDF uses word frequency scores that try to highlight words that are more frequent in a tweet but not across tweets.

3.1.2 N_gram modelling_word grams

To these 2 variants of bag of words methods we can add the N_gram modelling_word grams to take into account N_grams which increase the accuracy, as we assessed on our simulations.

3.2 Word2vec

We used embeddings pretrained on an external database or we trained vectors on our database. This model preserves the semantic relation based on the association of words (which words appear together), so it's an improvement for the bag of words modelling although it does not perform better when it comes to classification.

3.3 Glove

Global vector model is an improvement for the two previous modelling methods it's does not consider only local property of our data set but also the global property.

3.4 Elmo

To go further, Elmo embeddings are learned from the internal state of a bidirectional LSTM. Elmo

has the specificity to associate not only one vector to one token but as many tokens as the token's occurences. In each context, the token is represented differently. This method enables to get higher accuracy (around 80%) and to tackle polysemy on words such as 'fire' which can be used in disastrous tweets as well as in lighter tweets.

3.5 Model training

3.5.1 Classification with a naive approach using sentiwordnet

The idea is to say that predicting a disaster has something to do with sentiment analysis. More positive tweets may be irrelevant and more negative ones may point out a disaster.

3.5.2 Using classic classifiers

We used cross validation through pipelines to get the optimal parameters both for vectorization and for classification.

Pipeline W2V + classic classifiers: we tried support vector machine, random forest, logistic regression and stochastic gradient descent for classification with word2vec vectorization. The result is that it works better when the embedding is made on our database and it works well with random forest.

Pipeline CV + classic classifiers: we used support vector machine, random forest, multinomial naive bayes, logistic regression and stochastic gradient descent for classification with Countvectorizer. The good thing about these pipelines is that features are words (and not vectors in an abstract embedding space). Therefore, the most informative features can be interpreted: RandomForest seems to use more frequent words to discriminate tweets whereas LinearSVC relies on more specific words.

3.5.3 Vote classifier

We build a classifier which is the result of the vote between our previous classifiers.

3.5.4 Using deep learning classifiers

ELMO+NN Elmo is used here in a simple feed-forward neural network with one hidden layer with 256 neurons. Even with a simple architecture, it performs quite well on our data.

GLOVE+LSTM We are using LSTM which is a version of recurrent neural netword and recurrent neural netword are very useful for sequence learning and we can interpret the language at the sequence because every sentence is a set of words and the words appear in their definite order so it's a kind of sequence classification problem and recurrent neural networks are greater at that.

4 Results

We favour two metrics to interpret our models:

Accuracy: the accuracy is simply how good our machine learning model is at predicting a correct class for a given observation.

Recall: it calculates the part of actual disasters our model captures through labeling it as relevant (True Positive). We know that recall shall be the model's metric we use to select our best model when there is a high cost associated with False Negative which is the case with our problem.

Pipeline	accuracy	recall
('W2Vec', 'RF')	70.5	55.3
('CV', 'Li')	72.8	62.7
('CV', 'RF')	74.0	55.1
('CV', 'MNB')	75.7	65.3
('CV', 'LReg')	73.8	62.5
('CV', 'SGD')	73.0	61.9
Vote classifier	75.3	54.6
Sentiment score	47.2	60.3
Elmo + NN	81.5	77.5

5 Discussion/Conclusion

The marginal gain in accuracy using very sophisticated model is not so clear. Countvectorizer combined with classic classifiers as MNB or the vote classifier already performs quite well. Recall seems to be more sensitive to this sophistication.

Cross validation was possible for static words' embedding and for classic classifiers. If we had time to deal with computational costs, it would have been interesting to test this with deep learning models as well.

.