Natural Language Processing with Disaster Tweets

A classification task based on text

Jacopo Magliani

jacopo.magliani@studenti.unipd.it

1 INTRODUCTION

Natural Language Processing is a subfield of linguistic and computer science focused on the interactions between computers and the human language. The objective is making a computer capable of understanding the contents of a document and extrapolating valuable informations. Since a computer does not have the ability to directly interpret human language and in particular metaphors, preprocessing plays an important role in trasforming text into numerical data usable by the computer.

2 DATASET

The dataset had been provided by Figure Eight Incompany on their website "Data For Everyone" but in this project it has been downloaded from Kaggle website "Natural Language Processing with Disaster Tweets" [1]. The goal is predicting wich "tweets" (messages on the social network "Twitter" of less than 140 characters) are about real disasters or not.

The original dataset consists of 7613 instances and 5 features:

1. id (integer type)
2. keyword (string type)
3. location (string type)
4. text (string type)
5. target (binary integer type)

"id" is the primary key for each instance with no other use, "keyword" is considered as the most important word of the tweet and can be a single word (e.g. "earthquake") or a combination (e.g. "buildings%20burning"), "location" is the location entered by the author of the tweet, "text" is the script of the tweet, "target" is a binary value ([0,1]) to indicate if the tweet is about a danger (''target'' = 1) or not ("target" = 0).

3 METHODS

The followiing subparagraphs report the crucial points regarding the analysis of the content of the dataset and the necessary changes made to extrapolate usable information by the computer.

3.1 NAN VALUES

Looking for the presence of NaN values and how they are distributed among the features, it turns out there are 61 NaN values for "keyword" and 2533 for "location", which means it will be necessary to get the location of the tweet from the text. Since the instances with NaN values in "keyword" are less than 1% of the total dataset, they are erased from the dataset.

3.2 PREPROCESSING

As mentioned in the Introduction, preprocessing is fundamental before using machine learning models when dealing with text.

The basic steps were:

1. Cleaning a string of unnecessary characters.
2. Obtaining the location of the topic.
3. Obtaining new features.
4. Obtaining relevant words from the tweet.
5. Assigning a value to the keywords of a tweet to understand if it is about a danger.

3.2.1 CLEANING

Strings of "location" and "text" are cleaned of numbers, predefined special words (e.g. "www") and characters and common english words (e.g. "in", "we").

3.2.2 LOCATION

The location of the danger a tweet is talking about is logically mentioned in the text of the message, while sometimes the "location" feature mentions a place not bound to the locality of the danger (e.g. the instance with "id": 7918 has "location": "United States of America" but the "text" field is talking about a rainstorm in the Italian Alps).

GeoText package [2] was used to get the ISO country code of the topic of the tweet from the mentioned cities or states first by the feature "text", then by "location" if necessary (e.g. "Italy/Padova" -> "IT").

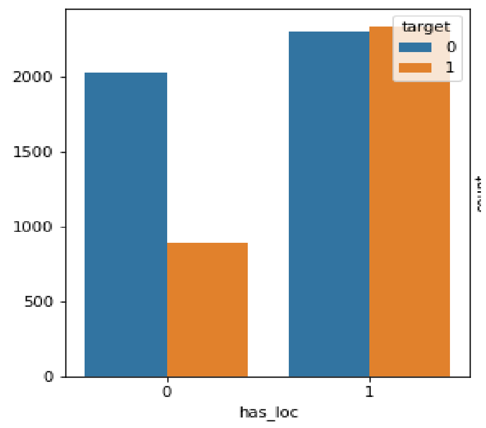
Pycountry package [3] was used to directly find the ISO code of the country of the danger first by the feature "text", then by "location" if necessary (e.g. "fire in IT" -> "IT").

Neither Geotext nor Pycountry can identify the american states as "US" so it was necessary to define a function looking for the mentions of the case (e.g. "fire in Alabama / AL" -> "US").

Observing some of the instances where it was not possible to retrieve the location of the topic of the tweet, indeed there were no useful mentions of it both in "text" and in "location" (e.g. "id":4356, "location": "One World").

3.2.3 ADDITIONAL FEATURES

Some additional binary features were defined to add complexity to the dataset indicating if all the words of the text are uppercase, if the location was successfully retrieved and if it was one of the most commons of the dataset and which of these.



Graph 1: histogram for the presence of location

3.2.4 RELEVANT WORDS

Since it is not practical to use different words that share the same root, stems and lemmas have been obtained to improve the search for similarities between different tweets.

3.2.5 IMPORTANCE OF WORDS

It is not convenient to use a method like pandas.get_dummy [4] to turn every categorial lemmas or stems into a binary feature to know if it is present in a tweet about a danger or not. Therefore it is calculated the ratio each word appears in a tweet about a danger on the total number of tweets it is used in.

4 EXPERIMENTS

These classification models were trained according to these proportion of the dataset (training set size: 0.8, validation set (from training set) size: 0.1, test set size: 0.2). For each model a range of parameters were tested by GridSearchCV [5] setting "accuracy" as scoring to evaluate the performance and 5 as the number of folds for the cross validation:

- Decision Tree Classifier
  - Criterion: gini, entropy
  - Max depth: [1,10,by=1]
  - Min samples split: [2,10,by=1]
- Random Forest Classifier
  - Criterion: gini, entropy
  - Max depth: [3,10,by=1]
  - N estimators: [10,100,by=10]
- K-Nearest Neighbors Classifier
  - N neighbors: [3,10,by=1]
  - Weights: uniform, distance
  - Metric: euclidean, chebyshev, minkowski, manhattan
- Multilayer Perceptron
  - Activation: logistic,relu
  - Solver: sgd, adam
  - Hidden layer sizes: (20,10),(20,20),(20,30)

It was decided to use the accuracy score as metric since the entropy target feature is high (0.98). The ideal parameters chosen for each model by GridSearchCV, the accuracy on the various sets and the average error on the test set and its components are below reported.

DECISION TREE CLASSIFIER (DTC): the best results were obtained with this combination of parameters:

- Criterion: entropy
- Max depth: 7
- Min samples split: 10

Train Acc	Validation Acc	Test Acc
0.78	0.74	0.76
Avg Exp Loss	Bias	Variance
0.245	0.234	0.080

RANDOM FOREST CLASSIFIER (RFC): the best results were obtained with this combination of parameters:

- Criterion: entropy
- Max depth: 9
- N estimators: 70

Train Acc	Validation Acc	Test Acc
0.81	0.77	0.77
Avg Exp Loss	Bias	Variance
0.232	0.221	0.043

K-NEAREST NEIGHBORS CLASSIFIER (KNNC): the best results were obtained with this combination of parameters:

- N neighbors: 10
- Weights: uniform
- Metric: euclidean

Train Acc	Validation Acc	Test Acc
0.79	0.74	0.76
Avg Exp Loss	Bias	Variance
0.252	0.234	0.090

MULTILAYER PERCEPTRON (MLP): the best results were obtained with this combination of parameters:

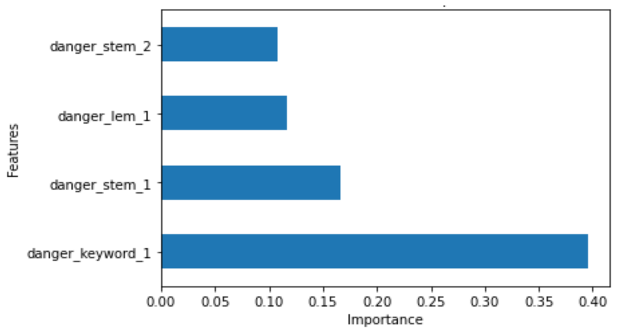
- Activation: relu
- Solver: adam
- Hidden layers sizes: (20,30)

Train Acc	Validation Acc	Test Acc
0.77	0.76	0.77
Avg Exp Loss	Bias	Variance
0.238	0.234	0.045

5 CONCLUSIONS AND RESULTS

From the table of the agglomerated results on the test set it can be seen that the models have obtained similar scores, with the Random Forest Classifier slightly better than the others, including the percentage of true positive (TP) and true negative (TN) classifications.

Model	Test Acc	Test Loss	TP-TN
DTC	0.76	0.245	29%-47%
RFC	**0.77**	**0.232**	**29%-48%**
KNNC	0.76	0.252	26%-49%
MLP	0.77	0.238	29%-47%



Graph 2 Feature importance for RFC

In addition, looking at the graphic representation of the importance of the features in the RFC it appears the relevance of the first keyword was dominant.

This suggests that the location of the topic of a tweet may not actually be useful for the classification task and that it is necessary to improve the preprocessing phase to be able to extrapolate the appropriate key term in the tweets where it was not already present.

6 REFERENCES

[1]https://www.kaggle.com/c/nlp-getting-started/overview

[2]https://pypi.org/project/geotext/

[3]https://pypi.org/project/pycountry/

[4]https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html