Using Tweets to Find Missing People in California Campfire

Team members: Shirui Chen, Zexin Du, Yanxi Long, Ruijie Wang, Zihao Yan, Qingchan Zhu

Repository: https://github.com/Christy0216/ISE-540/tree/Final_Project

Table of Contents

Executive Summary

The 2018 California Campfire was the deadliest and most destructive wildfire in California's history, and the most expensive natural disaster in the world in 2018 in terms of insured losses. It caused 103 confirmed fatalities, and over 10,000 houses were burned. The deadly wildfire in northern California in November 2018 was caused by power lines. Some technologies could be used for human beings to better find the useful information they need. In this project, what we want to do is to access related tweets of 2018 California wildfire by using Hydrator and see whether the social media like Twitter cooperated with nowadays' technology could be a helpful tool, positively affecting rescue process. Our team goal is applying machine learning models on tweets to develop an ideal algorithm for fire departments to find urgent and relevant messages from massive data.

Among this project, we implement various techniques. We apply Term Frequency and Inverse Document Frequency (TF-IDF) to vectorize texts and stemming techniques to change all future tenses and past tenses to the original form, such as did, has done, will do, are all considered as do. In addition, since we do not have labels from raw data, we manually label a small part of

tweets and use the labeled data to predict the rest of the training set by applying semi-supervised learning technique. Finally, we compute confusion matrix as a metric to measure the performance of models. Based on the algorithm we developed, we achieved 57% precision rate, 50% recall rate, and 0.53 F-1 Score. Even though the raw data is highly imbalanced and has massive noisy data. In addition, to deal with large and massive data we have, we introduced user-defined ensemble method and vote mechanism. The model we developed successfully helps fire departments to search for useful and relevant information, albeit with a room for model to optimize.

Propose Approach

Approach Overview

As discussed in the Executive Summary, our approach to this project consists of multiple steps and machine learning techniques. Followed is a basic outline of our project approach:

- Initial Data Collection
- Text Cleaning (Filtering, formatting)
- Data Preprocessing (labeling, vectorization, normalization, train/test splitting)
- Model Selection
- Deduce y Label using 8 Different Classifiers

Data Collection

Our data primarily came from different resources. CrisisNLP.org provides us with most of the tweet's information as well as missing and found list that are related to 2018 California wildfires

crisis. We also went on Twitter to extract even more related tweets to expand our database and increase resource completeness.

Text Cleaning

We have gathered more than 4 million pieces of data from both the dataset provided by CrisisNLP and from Twitter itself. The extraction process includes comparing lost and found list with raw dataset to find tweets with high relevance, filtering tweets with hashtags that contain missing/found etc., filtering tweets with valid addresses or dates of incidents, subtracting tweets with any name, descriptions, locations or any valid personal information, excluding repeating or reposting tweets and so forth. Finally, we have successfully retrieved around half a million valid data to assist our further analysis.

To ensure the quality of our data, we further our cleaning process with the following procedures. We lowercase all our data pieces and remove punctuation, numerical values, stop words, emojis. Then we replace similar words with affixes for better comparisons and analysis.

Data Preprocessing

After cleaning, we move forward to our data preprocessing step. Since the data is highly unorganized, we decide to use the following system to distinguish our data into two major categories:

Label data piece with 1 for urgent information: this type of data includes information highly relevant to missing or found personnel in the 2018 California wildfire crisis such as name of the victims, keywords of locations of the incident reported and current status (still missing/found/need help) etc.

Label data piece with 0 for less emergency data: this type of data includes updates on ongoing situations, broad range assistance requests, pets missing/found, properties damages or general comments etc.

We then randomly sample around 100k pieces of data and split them into training set and test set. We then encode tweets using TD-IDF to transform our texts to applicable vectors, and rescale and normalize our dataset, which is very essential to our further modeling and analysis. We also adopted principal component analysis to help us visualize data and find potential clusters and outliers. Finally, we randomly sampled 1200 pieces of information from our training set and manually label them using the categorizing system above for our initial training.

Model Selection

The choice of classifiers matters for every classification problem. Some algorithms are good at discrete variables, but some are very good at dealing with continuous variables. Some algorithms can perform on imbalanced data well, and some are very sensitive to the outliers.

As we know, most datasets of text analysis problems are linearly separable, so the reason we first choose logistic regression and naive bayes is that they can perform very well on the linear-separable data, also they are the two of most popular text classification algorithms because of independence of features and probability-based nature. They are sample and high-bias algorithms, so they can be trained faster than others.

We also tried tree-based algorithms and tree-based ensemble algorithms since they can also deal with text classification. From the metrics of those algorithms, we could see that decision-tree perform the worst among tree-based classifications because it is hard to control the depth of tree, which means it is easy to overfit and underfit. Then we tried random forest, gradient boosting

algorithm, and XGBoost. Three ensemble algorithms perform well on the training data because ensembling (bagging and boosting). Boosted and XGBoost are boosting based method, so the trees are built on the residual of previous trees, which is easy to overfit. But We used GridSearchCV trying to find the best parameters, which prevents the likelihood of overfitting.

We also tried SGDClassifier and support vector machine and performed cross-validation trying to find the best parameters, but their performance cannot beat the powerful ensemble method.

Deduce y Label using 8 Different Classifiers

We have selected 8 different classifiers to train our initial sampled data. We will review the accuracy of those models and make selections accordingly. The models selected and their best parameters are as follows:

- Random Forest

  {'max_depth': 100, 'n_estimators': 200, 'random_state': 0}
- Logistic Regression

  {‘class_weight: 'balanced'}
- Decision Trees

  {'max_depth': 100, 'min_samples_leaf': 7, 'random_state': 0}
- Gradient Boosting

  {'learning_rate’: 0.05, ‘max_depth’: 500, ‘max_features’: 'log2', ‘n_estimators’: 300}
- Naïve Bayes

  {'alpha': 0.1}
- SGDClassifier

  {'alpha': 0.01}
- XGBoost

  {'learning_rate’: 0.0, ‘max_depth’: 300, ‘n_estimators’ : 60}

- Support Vector Machine

  {'C': 0.1, 'gamma': 1, 'kernel': 'sigmoid'}

Experimental Results

After getting the parameters of all the models by using cross validation, we begin to run each of them and observe their performances. The accuracy of each model on the training set is shown in the following table.

Model	Accuracy Percentage
Random Forest	0.993
Logistic Regression	0.944
Decision Tree	0.943
Gradient Boosting	0.993
Naive Bayes	0.884
XGBoost	0.993
SGDClassifier	0.873
Support Vector Machine	0.892

According to the above table we can see that Random Forest, Gradient Boosting and XGBoost have the highest accuracy percentage, maybe because these three models can excellently deal with imbalanced data. Logistic Regression and Decision Tree have relatively lower

performances, but their accuracy percentage is still greater than 0.9. Naïve Bayes, SGDClassifier and Support Vector Machine perform worst.

After that, we use all the classifiers to create a voter. If four or more classifiers predict 1, we regard the final label of the data to be 1. Else we label the data with 0. Ultimately, we get 71291 1s and 8090 0s.

Next, we decide to use the labeled train set to train again, and then Manually label 800 random sample, which can greatly remove the negative influence brought by the imbalanced original data, from test set and compare them with the predict label. Then we use six classifiers, including Random Forest, Logistic Regression, Decision Tree, Gradient Boosting, Naive Bayes, SGDClassifier to create the new voter. (We do not choose to use XGBoost and Support Vector Machine because they run too slowly and we need to spend an exceedingly long time running them, though XGBoost does a good job on the training set.) Since our goal is to find information about campfires as much as possible, we need to catch each data that is reported to be "positive" by any classifier. Based on the performance of classifying as 1 of each classifier, we have decided our user-defined metrics to be:

If any classifier predicts 1:

Label 1

Else:

Label 0

The F1 – Score of each model on test set is shown on the following table.

Model	Accuracy Percentage
Random Forest	0.011494

Logistic Regression	0.518072
Decision Tree	0.433824
Gradient Boosting	0.0
Naive Bayes	0.0
SGDClassifier	0.188482

According to the table, we can observe that Logistic Regression has the highest F1 Score and Decision Tree has the second highest F1 Score. In contrast, Random Forest, Gradient Boosting and Naïve Bayes have the lowest F1 Score.

The confusion matrix is shown below.

	0	**1**	**Sum**
0	528	99	627
1	74	99	173
Sum	602	198	800

From the confusion matrix, we can see that there are 99 ones correctly labeled as 1, and 528 zeros correctly labeled as 0. Also, there are 99 ones that are mislabeled as 0 and there are 74 zeros that are mislabeled as 1.

We regard 1 as a positive case and 0 as a negative case. So True Positives, False Positives, False Negatives and True Negatives are 99, 74, 99, 528, respectively. The precision rate of the model is 0.57 and the recall rate is 0.50. The F1-Score of the model is 0.53.

The precision rate and the recall rate are nearly half, which means when we try to predict those tweets that are related to campfire, the model doesn't perform very well. Maybe it is because the data is imbalanced. Though the model does a good job when predicting True Negatives, considering that True Negatives are not so meaningful in our research, we can't be satisfied with this model.

Discussion

- The key metrics we choose to evaluate is F1-Score.
- Since the dataset is extremely imbalanced and highly possible to overfit, we introduce the voting mechanism to predict y labels.

Why do we use F1-Score?

F1-Score, which is used as the way of measurement in the analysis, is the harmonic mean of the value of recall rate and the value of precision rate.

$$F1 = \frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right)$$

The advantage of using F1-Score is that, if F1-Score isn't low, it guarantees that both the precision rate and the recall rate isn't too low. Only when these two rates are all high enough can the value of F1-Score be high. In our project, a high F1 Score not only guarantees that we can correctly find positive cases in all the tweets, but also can make sure that more positive cases can be found.

Why do we use voting mechanisms?

The reason we adopt the voting mechanism is that we would like to accurately compute the y label and increase our model accuracy but using only one or two classifiers for predictions would hardly bring us the desired results. Due to the dataset is highly imbalanced, some complex and high-dimensional models are prone to overfit in the training set. To solve this problem, we introduce the voting mechanism. Specifically talking, we adopt all classifiers instead of just one to label our data, and only label y when the majority of the classifiers (4 classifiers) predict it to be 1. We believe that by performing this voting mechanism, we could have a more precise labeling on our data, and thus higher F scores on the overall performance.

Conclusion And Future Work

Challenges

The first challenge of our project is that the total amount of data is pretty large and the data we have is extremely imbalanced, so we must take care of it very carefully. What's more, most of the data are kind of personal statements or opinions about the fire department, government, or some criticism to famous person, such as Donald Trump and Kenye West. Therefore, the first challenge is to filter some of them and manually label them as our training data. Another challenge of our project is that it is hard to define which kinds of tweets are positive. For instance, some tweets talk about the number of missing and dead people and some tweets talk about the number of missing pets. So, we spent plenty of time arguing and labeling the true positive case. Because of these 2 challenges, we did have relatively lower metric score on testing dataset, with F-1 score only about 53%.

Conclusion

On this project, as you could see we tried many different algorithms, and we have tuned many hyperparameters like “max depth” and “max features” in tree models, also kennel in SVM, and regulation on logistic regression. We found the best performance belongs to the random forest and gradient boost. Overall, the data is noisy, imbalanced, large, and massive, and needs to be labeled by hand, which is time consuming. The most challenging part of it is to manually label the positive. We tried to make it balanced, but it is impossible to do so since there must not be a lot of useful information on a social network. Therefore, we performed more than 5 machine learning algorithms and tried to do a user-defined ensemble, trying to catch every possible emergent and useful information from tweets.

Even though the model has a fair f-1 score, it could successfully save much time on finding the information from tweets. We can catch half of useful tweets from the whole massive tweets, and each one of two tweets we caught is useful for the fire, government or any other departments. Therefore, the model did a great job on it.

We highly recommend Twitter to set a special feature that only works for storing emergency tweets. We want Twitter to be a platform that truly helps after disaster happens. If it does so, there will be more useful information in a much less massive data lake, which makes researchists' lives easy, saving much time on cleaning and labeling massive data.

Future Works

Because of the large number of dataset and our device performance, we spent much time on running the code since there is not much useful information in just 100 thousand data. If possible, we could have a much more powerful device, we could try more models, such as

xgboost and support vector machine or even deep learning “Bert” on much more data, then we might have much more accurate prediction and higher metrics score.