Twitter has played an important role in social media life. Nowadays, people like to post Tweets to share things and many people get information from Tweets, like news, weather. For example, some people may post a Tweet when they see disasters to alarm other people. However, not every Tweet is about real disasters, sometimes people use words like “ablaze” metaphorically to describe the beautiful view.

To help people avoid some unnecessary harm from this ‘disaster’ information, we found a dataset from Kaggle which is related to this topic. What’s more, we attempt to use this dataset to build some machine learning models that predict which Tweets are about real disasters and which one’s aren’t.

In this report, we will cover following 6 parts to state this challenge: (1) introduction of the data;(2) data preprocessing; (3) 6 models used in the report; (4) evaluation of these models ;(5) conclusion and (6) reference.

**1, Introduction of Data**

The data is from Kaggle, and here is the link:

<https://www.kaggle.com/c/nlp-getting-started/overview>

There are three csv file: train, test and sample submission. For the train set, there are 7613 rows by 5 columns. each column represents id - a unique identifier for each tweet, keyword- a particular keyword from the tweet (may be blank), location - the location the tweet was sent from (may be blank), text- the text of the tweet and target-this denotes whether a tweet is about a real disaster (1) or not (0). For the test set, there are 3264 rows by 4 columns, which contains id, keyword location and text, with no target.

Since the target are only 1 and 0, it’s a binary classification problem. Our goal is using the train set to train some suitable models and predicting the test texts by our models.

**2, Data Preprocessing**

What we need to clean are some text contents. We checked the texts in our train set and found there are many noises in the text. Some of texts contains abbreviation and slang; Some of texts use special symbols like #, $, @ etc; what’s more, almost each of text has website information and uses punctuation. These are not conducive to our text analysis.

In the light of of the problems above, we provide 9 nine methods of data Preprocessing aiming to obtain the clean data: Mislabel, CONTRACTION\_MAP, @/#/http, Repetitive Letter, Abbreviation, Lowercase, punctuation, Stopwords and Keywords Variable.

**·Mislabel**

At the beginning, we grouped the train data by text and found there is a dozen of sentence have a different target, but the content of text are the same. Therefore, we a new column: target\_relabeled and revise the mislabel manually.

**·Contraction map**

Here, we use a contraction dictionary from the Internet to achieve some phrase like: ain't, don't, hadn't. we can use this dictionary to convert these phrase to their original form:is not, do not, had not. You can see the dictionary in the our github.

**·@/#/http**

In the train data, there are lots of website information, @ and # ,this part used for cleaning them.

**·Repetitive Letter**

Here, we write some loops to convert some words like 'loooook' , ‘goooood’ to 'look' ‘good’.

**·Abbreviation**

This part is different from contraction part. In English, there are some expression like 'lmao' means happy, 'icq' means 'I seek you'. To convert these slang to their normal form, we still downloaded a dictionary and a slang text. Then, we combine these two part to clean clean our text.

**·Lowercase**

This part we are going to uniform the case of all of our text content. We convert all of uppercase letter to lowercase.

**·Punctuation**

This part used for deleting all of punctuation of the text. We think that punctuation is sometimes meaningless when we do data analysis.

**·Stopwords**

In our text analysis, there are many words which didn't have actual meaning, like: 'to','at' etc. Here, we use the package **nltk.corpus**, which contains many stopwords to clean the text.

**·Keywords Variable**

Finally, in the training data, there is a column called keywords, it contains the disaster words of the text or the disaster this text described (didn't contain in the text). Since these words are important, we add them to the text.

**3, Bert as service: converting text to vector**

After prepocessing the text, we will encode the text content to a vector with specific length to run our traditional machine leaning model.

Our methods is bert as service. Now, what is bert as service? Simply speaking, bert as service is the encoder of bert. As we know, bert is a powerful deep learning model, which can solve nearly all of NLP problem. Therefore, we use bert as service to achieve our text to vector process.

Here is a link of bert as service tutorial bert as service:

<https://github.com/hanxiao/bert-as-service>