107061528 黃志翔

1. Preprocessing

Because the original data is stored as a json file, we need to load the json data and extract the wanted part first. Then according to the data identification csv to split the dataset into training and testing parts.

The information I used here is only the text part. Then I use the function below to clean and modify the original text tweets. (I modified the function from here) This function does the following things:

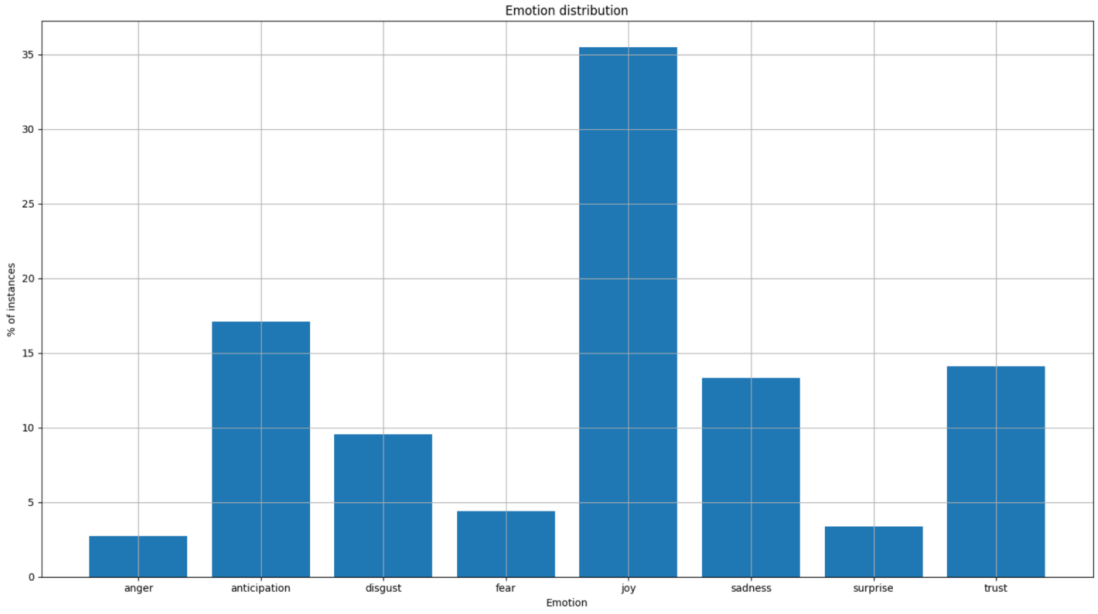
- Remove punctuation, url, number and hashtag
- Substitute the emoji by text description (use the package called demoji)

```
def spacy_cleaner(sentence):
    # Remove <...>
    sentence = re.sub('<.*?>', '', sentence)
    #
    emo = demoji.findall(sentence)
    if len(emo)>0:
        for i in emo.keys():
            sentence = sentence.replace(i, emo[i])
    apostrophe_handled = re.sub("’", "'", sentence)
    expanded = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t in apostrophe_handled.split(" ")])
    parsed = nlp(expanded)
    final_tokens = []
    for t in parsed:
        pdb.set_trace()
        if t.is_punct or t.is_space or t.like_num or t.like_url or str(t).startswith('@'):
            pass
        else:
            if t.lemma_ == '-PRON-':
                final_tokens.append(str(t))
            else:
                sc_removed = re.sub("[^a-zA-Z]", '', str(t.lemma_))
                if len(sc_removed) > 1:
                    final_tokens.append(sc_removed)
    joined = ' '.join(final_tokens)
    spell_corrected = re.sub(r'(.)\1+', r'\1\1', joined)
    return spell_corrected
```

After this kind of preprocessing, we can then move on to the feature extraction and training part.

2. Feature investigation and training strategies

We can first plot the distribution of the label and found that "joy" takes the most part and it seems this dataset some kind of imbalance.



First I use tf-idf as features and use undersampling/oversampling to different categories due to the data imbalance. I tried XGBoost and LinearSVC as classifiers but I got scores under 40 on kaggle, so I give up using traditional ML classifiers. (LinearSVC is much faster than XGBoost and even got a better score)

For deep learning strategies I first tried GloVe word embedding method to embed the sentences and then used bidirectional GRU as the model. But it did not improve the performance much, I think it is because twitter words are very fresh and many new words leads to that embedding is hard to implement.

At the end I tried the strategy in Lab2 (tf-idf + dense layers) then got a better score than what I did before……. Because it takes too much time for training, I give up modifying the model or tuning the parameters and turn to do the final competition…

One thing should be noted is that at first I do 3-fold CV to evaluate the strategies. But the dataset is too big to handle with limited resources so I use the result from kaggle score to modify the structures which is not good because I just see the kaggle score to decide whether to modify, which is some kind cheating because tuning for the testing set.