Homework 3 Report

Fangzheng Guo (fag24@pitt.edu)

CS2731 NLP, Fall 2020

Introduction:

For this homework, I used Jupyter Notebook with Python 3.7. Please see code blocks and comments in the notebook. The report includes description of my approach, experiment result and analysis for each step.

Task 1:

I split the dataset to 80% training set and 20% testing set. Since in origin the comments remain their order in the article, we should randomize the data to avoid bias. Because 2 adjacent comments tend to have the same property (is constructive or is not constructive). For cross validation, I split the training set to 5 folds.

Task 2:

For preprocessing, firstly I used regex to only keep the characters and numbers in the comment. Then I remove the stopwords (for example, "the", "a"…) in the comment since they are meaning-less. I also do stemming for all words, to reduce words with similar meaning in our vocabulary. Both removing stopwords and stemming steps are optional.

I used sklearn CountVectorizer to build up BOW vector for training set. According to 5-folds cross validation, the average accuracy of majority model is 0.525, the average accuracy of logistic regression model is 0.839 (without removing stopwords and stemming). We know that the logistic regression model performs better than the majority model (p-value: 0.00000001319950393971).

Task 3:

Sparse vectors are generated by TF-IDF method (sklearn TfidfVectorizer). I used pre-trained word vectors (Standord GloVe, 6B tokens, 400K vocab, uncased, 50d vectors trained on Wikipedia 2014 + Gigaword 5) to get the dense vector for each comment in the training set (after experiment, I found training our own model with such a small size of data is effect less).

In a comment, for the words appeared in the comments which are not in the pre-trained word vector set, I will skip it. For the other words founded in the set, I will sum their vectors up and finally divide the sum by the number of these words (get the average vector). The result will be the dense vector for this comment.

Experiment result for BOW, TF-IDF, word2vec:

Cross validation accuracy	BOW	TF-IDF	Word2vec
No preprocessing	0.8392	0.85961	0.6857
Remove stopwords	0.8177	0.7756	0.6581
Stemming	0.8440	0.85959	0.6329
Remove stopwords + stemming	0.8380	0.7960	0.6077

From the result we know that, TF-IDF model without removing stopwords and stemming has the best performance. Its accuracy on testing set is 0.799. For large dataset, word2vec model tends to have the best performance. However, since we don't have a significant amount of data, word2vec model's performance is much lower than BOW.

Here we can come up with a question, does removing stopwords and stemming really helps in this specific case? The answer is no. The result shows that stemming does not have benefit, one possible reason is our vocabulary is relatively small. Moreover, removing stopwords will make the cross-validation result become worse. The reason is, the length of the comment (number of words) matters (intuitively a longer comment tends to include more information and has a bigger chance to be objective and constructive). Removing some words from a comment will change its length.

Task 4:
My question is: will a balanced training set helps to improve the model's performance?

In experiment I used imblearn RandomSampleOver to balance the training set.

Result (5-fold cross validation, without removing stopwords and stemming)

	BOW	TF-IDF	Word2vec
imbalanced	0.8392	0.85961	0.6857
balanced	0.8596	0.8652	0.6745

TF-IDF model's performance on testing set is now 0.7895.

From the experiment result, we know that balancing our dataset will not be very beneficial to model's performance. The reason is, our origin dataset is balance (majority model's accuracy is 0.525 proves that). As a result, there is no need to rebalance the training set.

The question about whether the preprocessing steps are necessary or not is addressed in Task 3 section.