Assignment 3 Report

November 4, 2019

1 Preprocessing

The excel file contains 12 columns, but I only kept "comment_text" and "toxicity_level", since they are the input and output of our task. Some columns such as "is_constructive" might be useful. However, since this is a course assignment with a small and skewed corpus, I would like to keep it simple. As a result, those columns were dropped.

1.1 comment_text

For each cell of the comment text, I did a normal text preprocessing:

1) Removing punctuation and white spaces, since they are pretty common in every text and not indicative; (except "?" and "!", but these two are limited in this corpus)

2) Removing stop words. The reason is similar to the previous one, because stop words such as "a", "the", and "to" also do not represent the core information of a text. NLTK has a set of stop words, and we can access it by `stopwords.words('english')`;

3) Performing a simple lemmatization on nouns. Because the NLTK lemmatizer cannot recognize the POS tagging of a token, the default parameter for the lemmatizer is noun.

1.2 toxicity_level

According to one of the official document from SFU, toxicity means the offensive level of a text. The level ranges from 1 to 4. The higher the level is, the more offensive a text is. In the original excel file, each cell of column "toxicity_level" may contain more than one result, and each result corresponds a confidence score. The result with the highest confidence score is called ground truth. In order to maximize the accuracy, I only kept the ground truth, and those labels with low confidence scores were removed.

1.3 K-fold cross validation

The distribution of toxicity level labels are considerably skewed – there are approximately 80% texts with label "1". If the original k-fold cross validation from sklearn is used, each chunk may not be able to hold the original label proportion, which will create an inaccurate environment for training and testing. Therefore, I rewrote the k-fold cross validation function. The new function will split the pool of each label into k equal parts, so during each part of training and testing, there will be enough data of all labels.

2 Performance

2.1 Evaluation metrics

To evaluate the performance, I used recall/precision with micro average. The recall and precision will give identical score, since Recall = $\frac{TP}{\text{Actual Results}}$ and Precision = $\frac{TP}{\text{Predicted Results}}$, and both two results have same amounts of data. As for the calculation of average, my reason for choosing micro average is because of the skewed distribution. According to the documentation page from sklearn, micro average will "calculate metrics globally by counting the total true positives, false negatives and false positives", while macro average will "calculate metrics for each label, and find their unweighted mean." Since the data is skewed, it is obvious that the score of label "2", "3", and "4" will not be high, which means that the macro average could be much lower than micro average. Moreover, because we only care about global accuracy instead of the accuracy of a specific minor label, micro average is more suitable.

2.2 Result

For step 2, I used CountVectorizer for bag-of-words. For step 3, I used TF-IDF for sparse vector embedding and Word2Vec(KeyedVectors) for dense vector embedding. All of those models above, plus the logistic regression are imported from sklearn. Besides, the pretrained Word2Vec model is converted from a Twitter GloVe model, and the sentence/paragraph vector is calculated by adding all word vectors in it. I chose a pretrain model instead of training the Word2Vec model on the corpus, since the corpus contains limited vocabulary. As a reference, the majority vote score is 0.7967.

Model	Bag-of-words	TF-IDF	Word2Vec
k = 5	0.7767	0.7958	0.7948
k = 15	0.7707	0.7952	0.7952
k = 25	0.7694	0.7945	0.7954
k = 35	0.7739	0.7952	0.7951

Table 1: Recall (micro average) of each model

Basically the performance of bag-of-words is a little bit lower than majority vote, while the TF-IDF and Word2Vec models are approximately equal to the majority vote. Due to the time limit, I didn't do the test with a high value of k. (larger than 100) With a small value of k, TF-IDF should have a higher score than bag-of-words, since TF-IDF is calculated by multiplying the term frequency and the inverse document frequency, which could represent the feature of the original text better. What surprised me is that the Word2Vec model has similar score as TF-IDF. I guess that there might be a soft cap on 0.7967, which is the score of majority vote, but unfortunately I cannot prove it.

3 Extra Experiment

The score of these three models are not that impressive, so I thought that it might be because of the skewed corpus. As I mentioned before, there are about 80% texts are label "1", and only 20% texts are with label "2", "3" and "4". If we take this part separately, the number is even smaller. Therefore, I combined label "2", "3" and "4", and called them label "2". By using this new label method, the number of labels is reduced

to 2: "1" for texts that are not toxic, and "2" for texts that are toxic. (no matter how toxic they are) The performance of this new label method is in the following table (Only TF-IDF and Word2Vec are recorded because the performance of bag-of-words is worse than those two):

Model	TF-IDF	Word2Vec
k = 5	0.7929	0.7948
k = 15	0.7933	0.7952
k = 25	0.7945	0.7954
k = 35	0.7942	0.7951

Table 2: [New Label] Recall (micro average) of each model

I cannot see any improvement here, and again, this might be because of the soft cap. Since the dataset is skewed and the number of contexts are limited, models will be more likely to work like majority vote. The best solution that I can come up with could be some improvements on the corpus itself:

1) Enlarge the dataset;

2) Add more texts with high toxicity level and increase the proportion of label "2", "3" and "4".