NLP homework3 report

Shuchen Liu (shl174@pitt.edu)

Step1

At first, I randomly split the whole dataset into training set and test set. The training set is about 80% of the original data set and the test set is about 20%. Then I randomly split the training set into 5 folds to do cross validation later.

```
## read data from file
df = pd.read_excel("SFUcorpus.xlsx")
df['toxicity_level'] = df['toxicity_level'].apply(lambda x: str(x)[:1])
X = df['comment_text']
Y = df['toxicity_level']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

Step2

For the preprocessing, I tried three things. First, I only kept the letter and number in comments. Second, I tried to remove stop words from the comments. Third, I tried to do stemming for each word in comments. The preprocessing is done on both the training set and test set.

```
## pre-processing
stopwords = set(nltk.corpus.stopwords.words('english'))

## get rid of noise and stop words
def preprocess(x,stopwordsOrNot,stemmingOrNot):
    x = re.sub('[^a-z0-9\s]', '', x.lower())
    tmpList = x.split(' ')
    if stemmingOrNot:
        ps = PorterStemmer()
        for i in range(len(tmpList)):
            tmpList[i] = ps.stem(tmpList[i])
    if stopwordsOrNot:
        tmpList = [w for w in tmpList if w not in set(stopwords)]
    return ' '.join(tmpList)
```

After the preprocessing, I used the “CountVectorizer” in “sklearn” to build the bag of words vectors of training set. After that I used 5 folds cross validation to train and evaluate a logistic regression model. The average accuracy of this model on 5 test parts is 0.736 and the majority voting accuracy on this data set is 0.790.

```
## bag of words
vectorizer1 = CountVectorizer()
X_BOW_train = vectorizer1.fit_transform(X_train)
```

Step3

In this step, I first tried to use the tf-idf method to generate the sparse vectors of the training set. The “TfidfVectorizer” in “sklearn” is used here.

```
## tf_idf
vectorizer = TfidfVectorizer()
X_TFIDF_train = vectorizer.fit_transform(X_train)
X_TFIDF_res, y_TFIDF_res = oversamples.fit_sample(X_TFIDF_train, y_train)
```

Then I tried to use the “glove.6B.300d.txt” file as the pre-trained word vectors to get the dense vector of each comment in the training set. First, I read the file and save each word of this file into a dictionary. The key is each word and the value is the 300-dimension vector of each word.

Then for each comment, I would sum the vectors of each word in this comment and divide the result by the number of non-zero vectors to normalize the result. For example, a comment such as “very good book to read” would be “very good book read” after removing stop words. If “read” is not in the pre-trained word vectors set, the 300-dimension dense vector of this comment would be the sum of the vectors of “very”, “good”, and “book” then divided by 3.

If all words in a comment are not in the pre-trained word vectors set, it would be transformed as a 300-dimension zero vector.

```
## Dense vector
map = dict()
with open('glove.6B.300d.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split(' ')
        word = values[0]
        value = np.asarray(values[1:], dtype='float32')
        map[word] = value

## transform df to map
vectors = []
for i in X_train:
    vector = np.zeros(300)
    count = 0
    tokens = word_tokenize(i)
    for c in tokens:
        if c in map:
            count += 1
            vector += map[c]
    if count == 0:
        vectors.append(vector)
    else:
        vectors.append(vector / count)

X_W2V_train = pd.DataFrame(vectors)
```

Now I already had the training data represented as bag of words, tf_idf and dense vectors. I used the 5 folds cross validation to train and evaluation the logistic regression model and get the following results.

accuracy	Bag of words	Tf_idf	Dense vectors
No process	0.767	0.794	0.728
stemming	0.730	0.791	0.718
Removing stop words	0.754	0.792	0.705
Stemming + removing stop words	0.736	0.788	0.681

```
## cross_validation
## bag of word result
if balanceOrNot:
    X_BOW = X_BOW_res
    y_BOW = y_BOW_res
else:
    X_BOW = X_BOW_train
    y_BOW = y_train
scores_majority = cross_val_score(majority, X_BOW_train, y_train, cv=5)
scores_lgm_BOW = cross_val_score(lgm, X_BOW, y_BOW, cv=5)
print("bag of word input")
print("average accuracy of majority model: ", np.mean(scores_majority))
print("average accuracy of logistic regression model: ", np.mean(scores_lgm_BOW))
```

```
## tf_idf result
if balanceOrNot:
    X_TFIDF = X_TFIDF_res
    y_TFIDF = y_TFIDF_res
else:
    X_TFIDF = X_TFIDF_train
    y_TFIDF = y_train
scores_majority = cross_val_score(majority, X_TFIDF_train, y_train, cv=5)
scores_lgm_TFIDF = cross_val_score(lgm, X_TFIDF, y_TFIDF, cv=5)
print("tf_idf input")
print("average accuracy of majority model: ", np.mean(scores_majority))
print("average accuracy of logistic regression model: ", np.mean(scores_lgm_TFIDF))
```

```
## Dense vector
if balanceOrNot:
    X_Dense = X_W2V_res
    y_Dense = y_W2V_res
else:
    X_Dense = X_W2V_train
    y_Dense = y_train
scores_majority = cross_val_score(majority, X_W2V_train, y_train, cv=5)
scores_lgm_Dense = cross_val_score(lgm, X_Dense, y_Dense, cv=5)
print("dense vector input")
print("average accuracy of majority model: ", np.mean(scores_majority))
print("average accuracy of logistic regression model: ", np.mean(scores_lgm_Dense))
```

Since the dataset is not balanced, the performance of each model is not good enough. Only tf-idf models are better than the majority voting. According to the cross validation result, the best model is the tf-idf model with stemming, so I used it to predict the test set and get an accuracy of 0.808.

Step4:

I compared the result of each model with the majority voting baseline. Based on the t-test result, although the accuracies of tf_idf models are higher than the majority voting baseline, they are not statistically significant (with p-value 0.25, 0.20 and 0.68 which are larger than 0.05).

```
## statistical tests
## BOW t-test
stat1, p1 = ttest_ind(scores_lgm_BOW, scores_majority)
print("BOW p value: ",p1)

## Dense vector t-test
stat2, p2 = ttest_ind(scores_lgm_Dense, scores_majority)
print("dense vector p value: ",p2)

## tf_idf t-test
stat3, p3 = ttest_ind(scores_lgm_TFIDF, scores_majority)
print("tf_idf p value: ",p3)

## predict the test set
lgm.fit(X_TFIDF, y_TFIDF)
# X_TFIDF_test = vectorizer.transform(X_test)
X_TFIDF_test = vectorizer.transform(X_test)
y_predict = lgm.predict(X_TFIDF_test)
print(accuracy_score(y_test, y_predict))
```

Step5

My first question is whether stemming and removing stop words would improve the performance. Based on the test before, the accuracies with or without them are around 0.7 and they are not statistically significant. So, I think stemming and removing stop words are useful in NLP however since this dataset is small and imbalanced, they cannot improve it a lot.

I also noticed that with the stemming input, dense vectors did not perform well. I think the reason is after stemming, the word would no longer in the pre-trained word vectors set.

My second question is whether balancing the training set would improve the performance.

I suppose the performance of the model trained on this balanced set would be better however since the oversampling added to much data in the original training set, it might not perform well when some new input shows up.

In order to get a balanced data set. I used “RandomOverSampler” in “imblearn” to make a balanced training set.

```
## balance the training set
oversamples = RandomOverSampler(random_state=6)
X_BOW_res, y_BOW_res = oversamples.fit_sample(X_BOW_train, y_train)
```

Then I used 5 folds cross validation on logistic regression model with three different input features on this balanced training set. I found that the accuracy is extremely high as follow.

Bag of words	Tf_idf	Dense vectors
0.954	0.975	0.873

However, it is definitely overfitting, the accuracy on test set (0.77) proved it.

Here is the running result of my program

```
process 1: do noting
bag of word input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7674488684967727
tf_idf input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7949510070767556
dense vector input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7289368017212328
BOW p value:  0.005044859961877725
dense vector p value:  0.0001014271690476775
tf_idf p value:  0.2518200012084848
test acccuracy:  0.8038277511961722


process 2: stemming+stopwords
bag of word input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.736209010550328
tf_idf input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7889774995463619
dense vector input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.6810088913082926
BOW p value:  8.654062949755069e-05
dense vector p value:  8.122875755127097e-06
tf_idf p value:  0.6994734626534351
test acccuracy:  0.7990430622009569
```

```
process 3: stopwords
bag of word input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7541875988283173
tf_idf input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7925703139176192
dense vector input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7050053140472301
BOW p value:  0.012895014969929999
dense vector p value:  5.760361673835494e-07
tf_idf p value:  0.2083891143044148
test acccuracy:  0.7990430622009569


process 4: stemming
bag of word input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7302502786634524
tf_idf input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7913872255489022
dense vector input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.7182085180288773
BOW p value:  0.00047005361809015144
dense vector p value:  1.2244663196448943e-05
tf_idf p value:  0.6801631380980224
test acccuracy:  0.8086124401913876


process 5: stemming+balance+stopwords
bag of word input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.9548577376821651
tf_idf input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.974968193384224
dense vector input
average accuracy of majority model:  0.790182232936724
average accuracy of logistic regression model:  0.8732824427480915
BOW p value:  2.2875988596448456e-13
dense vector p value:  1.8657011736601324e-07
tf_idf p value:  1.8023343612882104e-11
test acccuracy:  0.7703349282296651
```

Summary

In this task, I used three different ways to extract features from comments and used logistic regression model to do the prediction based on the input features. However, the performance of each model is not good enough, just same as the majority voting model. I think the reason is that the dataset is small and imbalanced. I believe with a larger and balanced dataset and some other machine learning models such as random forest or RNN I might have a better model.