Sentiment Classification of Tweet Report

Anonymous

1 Introduction

Emotion classification is a very popular field in natural language processing. 160,000 sentences were given for labeling and categorizing, for predicting whether the emotion in the statement was positive, neutral or negative. The data is divided into training set, development set and test set. In the data label, the frequency and importance characteristics of some words in each sentence (excluding the extremely high frequency and extremely low frequency words) are provided, and 100 feature vectors are provided as semantic features of the sentence. In this paper, KNN classifier and multiple linear regression classifier are used as tools for emotion prediction. Because KNN algorithm is simple in idea, easy to understand, easy to implement, and does not need to estimate parameters, it is especially suitable for multi-classification problems. Multiple linear regression algorithm is simple, programming is convenient, calculation is simple, decision-making speed is fast.

2 Literature review

The two papers from the data set used different methods to classify the emotions of a large amount of user data, texts and even images generated in social networks. Go et al. [1] believed that naive Bayes, maximum entropy and support vector machine could predict the emotions of Twitter with high accuracy. Vadicamo et al. [2] argues that deep convolutional neural networks can effectively predict social media sharing content with text and images. In addition, other papers use a combination of multiple algorithms. Tripathy et al. [3], for example, see better accuracy in using support vector machine algorithms to select the best features of the IMDb and Polarity datasets and then using artificial neural networks to calculate precision values. Liu et al. [4] compared the emotion classification results of four feature selection algorithms and five machine learning training algorithms respectively, and believed that the gain ratio algorithm of the former and the support vector machine algorithm of the latter had the best classification accuracy.

3 Method

The model requires emotional classification of Tweets text data, which can be divided into three parts: positive, negative and neutral. The three data sets given were a training set with annotations and a development set for feature selection and debugging models, and a test set without annotations for final evaluation. In the code implementation section, the library Pandas and Numpy were introduced for data processing and the library Matplotlib introduced for drawing.

First, read the four files of train, dev and test respectively, count, full, glove, tfidf. Then the Tweets column (100 feature vectors) of train_glove and dev_glove

are processed and transformed from string form to list form suitable for Data Frame in Pandas. Check the vocab file, it can be seen that after removing the extremely high frequency and extremely low frequency words, there are 5000 words to be analyzed, which correspond to different times of occurrence in the text respectively.

Out[174]:

	0	1
0	oh	3083
1	sorry	4054
2	pets	3245
3	love	2663
4	prison	3419
...	...	...
4995	gods	1888
4996	volleyball	4719
4997	ut	4670
4998	chem	890
4999	noodles	3022

5000 rows × 2 columns

Figure 1: vocab.txt shown in Pandas DataFrame

After statistical mapping of sentiment column in train_glove and dev_glove, it can be found that negative and positive emotions account for almost the same proportion, while neutral words account for less, only about half of the first two.

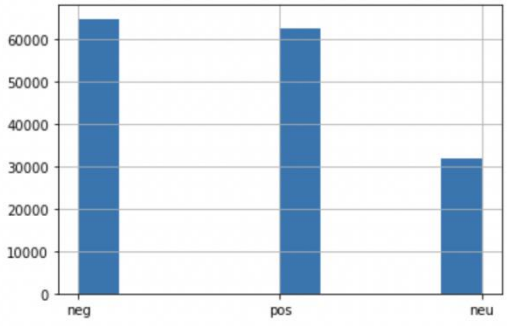

Figure 2: sentiment labels of training set

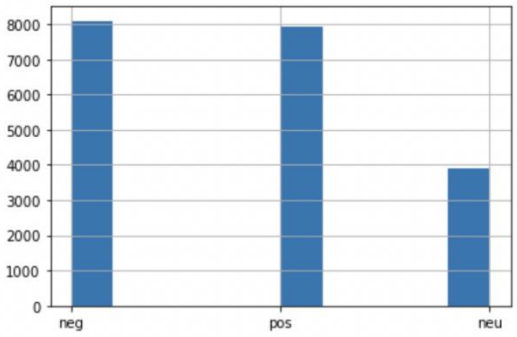

Figure 3: sentiment labels of development set

After observing the data set, several classifiers are introduced, including Naive Bayes, Decision Tree (set tree depth to none and 1 here), K neighbor (set neighbors to 1 and 5 here) and logistic regression model (set default solver parameter, 1 degree of random state and multinomial of multi_class make loss fit across the entire probability distribution.

Then briefly explain why choosing these classifier to train sentiment data. The Naive Bayes Classifier, less sensitive to missing data, has less computation, supports lazy learning and incremental learning, extrapolation is like table lookup, which is extremely fast. The Decision Tree Classifier has a small amount of computation, can clearly express the importance of attributes, and can be incrementally learned to partially reconstruct the model. It does not require knowledge in any field and parameter assumptions, and is suitable for high-dimensional data. The K Nearest Neighbor classifier is simple in idea, easy to understand and implement, having no parameter estimation and no training, is suitable for classifying rare events, especially for multiple classification problems. The Logistic

Regression model algorithm has relatively simple calculation, fast decision-making speed and small computation.

```
GNB :acc 0.4493620014066111
One-R :acc 0.4908067919220336
1-Nearest Neighbor :acc 0.5835426504571486
5-Nearest Neighbor :acc 0.6119762885562142
Decision Tree :acc 0.5429016376971767
Logistic Regression :acc 0.6788907866974782
```

Figure 4 : Accuracy of each Classifier

This report used the processed the train glove data set and it's sentiment labels to train different classifier models, and then used the development data set to calculate their accuracy. Through preliminary model training, it can be seen from figure 4 that the accuracy of Naive Bayes is the lowest, less than 50%, while the accuracy of Multiple Linear Regression is the highest, which can reach nearly 68%.

4 Results

After obtaining these data, this report further optimizes the above model. Firstly, for the decision tree model, the tree depth of baseline is set to 1, and the classifier is set without limiting the maximum depth. The results are then evaluated using the metrics module in sklearn.

```
In [184]: from sklearn.metrics import precision_recall_fscore_support
          from sklearn.metrics import accuracy_score
          precision_recall_fscore_support(dev_label, pred, average='macro')
Out[184]: (0.5606912898912966, 0.56129949204929, 0.5609938088821346, None)

In [185]: accuracy_score(dev_label,pred)
Out[185]: 0.5419973877222948

In [186]: dt_clf.get_depth()
Out[186]: 42
```

Figure 5 : Precision, recall and F-score of decision tree model

It can be found that the values of precision, recall and F-score are all about 56%, which are relatively stable and moderate. Precision value means 56% of the cases are correct when an instance is predicted, recall value means 56% of the truly sentiment instances have been correctly identified, and F-score is a metric to evaluate them as a whole, so that both of these two indicators are coordinated, nor too high or too low. In addition, the depth of decision tree in the results reaches 42, which is a relatively complex tree.

```
In [187]: acc1=[]
          for i in range(1,42,10):
              clf=DecisionTreeClassifier(max_depth=i,max_features='auto')
              clf.fit(train_glove_tweet,train_label)
              score=clf.score(dev_glove_tweet,dev_label)
              acc1.append(score)
              print(score)

0.46599015372249575
0.582889581030845
0.5372751934090224
0.5326032352054657
0.5296393047322415

In [188]: acc2=[]
          for i in range(1,42,10):
              clf=DecisionTreeClassifier(max_depth=i,max_features='auto')
              clf.fit(train_glove_tweet,train_label)
              score=clf.score(train_glove_tweet,train_label)
              acc2.append(score)
              print(score)

0.4348238337739 32
0.6086792713481065
0.9355679327861955
0.999204266167 67
0.9999058102516122
```

Figure 6 : Accuracy of each tree depth

Then take different tree depths, from 1,11,21,31 to 41. It can be seen that the accuracy rate is higher in the training set. When the tree depth reaches 31 and 41, the accuracy rate can reach 99.9%. The balance accuracy of the development set is about 56%, and the accuracy is also stable between the above different tree depths (47%-58%) under the assumption that the data sample classification is balanced. It can be seen that the classification equilibrium of emotional data itself has some influence on the final

accuracy, but the influence fluctuation is small.

```
In [190]: import matplotlib.pyplot as plt
          x=range(1,45,10)

          plt.plot(x,acc2)
          plt.plot(x,acc1)

Out[190]: [<matplotlib.lines.Line2D at 0x7f99246ed0a0>]
```

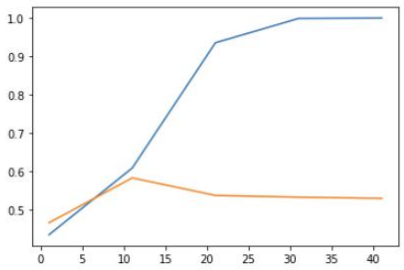

Figure 7 : Plot of two data sets' accuracy (decision tree model)

Then a graph is drew to observe the learning curve. As shown in figure 6, X is the depth of the tree, and Y represents the accuracy of the two data sets. The two colored curve represents training sets and development sets. It can be seen that on the left side of the figure, there is underfitting problem of the model, the two curves are too close to each other, and the prediction results have high bias. On the right side of the figure, there is the overfitting problem of the model, and the prediction results have high variance.

Then, this report further selects the features of the model by introducing the SelectKBest module and selecting only the best representative features of K. Since each id has 100 feature vectors, KBest is set to start with 10, 20, 40, 80 and 100 (all features), and the model is readjusted to obtain the accuracy for chi square statistics and mutual information respectively.

Figure 8 : Accuracy of each K Best Features

It can be seen that with the increase of the number of eigenvalues selected, the accuracy is gradually improved, rising from 56% to 67%. It can be seen that the more eigenvalues selected, the more conducive to the accuracy of prediction. At the same time, the accuracy of mutual information for the sentiment analysis data is better than chi-square statistics in most cases.

After completing the optimization of the decision tree model, this report continues to do further tuning for other models. Similarly, the report repeats the above steps and sets the baseline parameter as 1 and the classifier parameter as 5 for KNN model. As can be seen from Figure 9, the precision, recall, F-Score and accuracy values of the development set are generally superior to those of the previous decision tree model, all of which exceed 60%.

```
In [195]: from sklearn.metrics import precision_recall_fscore_support
          from sklearn.metrics import accuracy_score
          precision_recall_fscore_support(dev_label, pred, average='macro')

Out[195]: (0.6693712380703145, 0.6151361302058862, 0.6354565461760928, None)

In [196]: accuracy_score(dev_label,pred)

Out[196]: 0.6119762885562142
```

Figure 9 : Precision, recall and F-score of KNN model

```
In [197]: acc1=[]
          for i in range(1,10,2):
              clf=KNeighborsClassifier(n_neighbors=i)
              clf.fit(train_glove_tweet,train_label)
              score=clf.score(dev_glove_tweet,dev_label)
              acc1.append(score)
              print(score)

0.5835426504571486
0.5963026223249271
0.6119762885562142
0.6183060383803878
0.6230282326936603

In [198]: acc2=[]
          for i in range(1,10,2):
              clf=KNeighborsClassifier(n_neighbors=i)
              clf.fit(train_glove_tweet,train_label)
              score=clf.score(train_glove_tweet,train_label)
              acc2.append(score)
              print(score)

0.999830458452902
0.7846382799696081
0.7456688414033017
0.7236409989136783
0.7104669927725067
```

Figure 10 : Accuracy of each neighbors

Further, the number of neighbors are not limited, assuming them to be 1,3,5,7,9, respectively, the performance of the development set and training set are observed. It can be seen that when the number of neighbors is 1, the accuracy rate of the training set is 99.9%, but the performance of the development set is not ideal, which is in the worst of all conditions. In contrast, the worst case of training, with 9 neighbors, performed best in the development cluster, with 62 percent. Therefore, it is difficult to meet the high accuracy of development set and training set at the same time under a certain number of neighbors, which is a trade-off.

```
In [200]: import matplotlib.pyplot as plt
          x=range(1,10,2)

          plt.plot(x,acc2)
          plt.plot(x,acc1)

Out[200]: [<matplotlib.lines.Line2D at 0x7f9958351b80>]
```

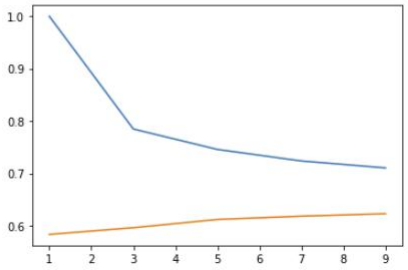

Figure 11 : Plot of two data sets' accuracy (KNN model)

This report present the above situation in the form of graph, as shown in figure 11. On the left side of the graph, overfitting problem occurs, and data has high variance; on the right side, underfitting problem occurs, and data has high bias. In the location with 3 neighbors, there is an obvious inflection point. Therefore, it can be considered that when the number of neighbors is equal to 3, the fitting of the model is relatively in high quality.

```
In [95]: from sklearn.metrics import precision_recall_fscore_support
         from sklearn.metrics import accuracy_score
         precision_recall_fscore_support(dev_label, pred, average='macro')
Out[95]: (0.6901028174089197, 0.6985866560016424, 0.6941097505741629, None)

In [96]: accuracy_score(dev_label,pred)
Out[96]: 0.6773837034060083
```

Figure 12 : Precision, recall and F-score of logistic regression model

This report further optimize the logistic regression model. Firstly, it can be seen that the precision, recall and F-score values of this model are better than those of KNN model, and generally reach 69%. The balance accuracy score was close to 69.9%. Here, the maximum number of iterations of Solver convergence under this model is tried to be

optimized. When the default value is 100, 0, 50, 100, 150 are taken respectively to observe the results.

```
In [57]: import matplotlib.pyplot as plt
         x=range(0,200,50)

         plt.plot(x,acc2)
         plt.plot(x,acc1)
Out[57]: [<matplotlib.lines.Line2D at 0x7f93d04f8790>]
```

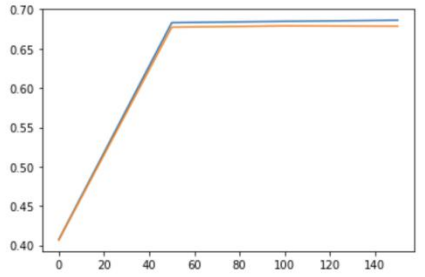

Figure 13: Plot of two data sets' accuracy (logistic regression model)

It can be found that the accuracy of multiple linear regression has similar fitting degree in different ranges. When the maximum number of iterations is more than 50, the accuracy is basically optimal, basically maintaining at about 68%, however, with the increase of the number of iterations, bias also increases slightly. When the number of iterations is less than 50, the bias is relatively low, but the accuracy is not ideal.

Finally, this report uses Bagging scheme as a remedy for the overfitting problem in the decision tree model. Bagging method combines multiple models and randomly selects training data to make prediction results more stable and reduce variance of single model. Here, the maximum depth is set to 11 (the optimal depth trained previously), the number of predictors to 3, and the maximum sample proportion to 30%. The accuracy of the results obtained reached 59%.

5 Discussion/Critical Analysis

I think there may be bias in data selection and analysis. For example, data are selected from several papers, however, whether the comments on twitter come from certain topics, whether the comments are made at the same time, where the users of the comments come from, what age group they belong to, and what occupation they are, all these factors may affect the model construction and optimization. Another example is that the mood classification of each Tweet has been marked in the given training set and development set, but it is not necessarily accurate, because there may be subjective factors of human judgment.

To do this, new features can be added to the dataset. First, use the RE library to remove meaningless characters from the text, such as the user name after @, the number string after #, and the web address after HTTPS, which are completely irrelevant to sentiment analysis. Then, textblob library is introduced with sentiment. Subjectivity and sentiment. Sentiment can be used to measure the subjectivity and polarity of emotional tendencies in texts. As figure 14 shows, 150,000 lines of text have successfully added new eigenvalues -- polarity and subjectivity.

```
In [139]: new_feature
```

Out[139]:

	Polarity	Subjectivity
0	-0.500000	1.000000
1	0.250000	0.800000
2	0.600000	1.000000
3	0.375000	0.200000
4	0.016667	0.633333
...	...	...
159248	0.000000	0.500000
159249	0.437500	0.650000
159250	0.437500	0.300000
159251	0.000000	0.125000
159252	0.000000	0.000000

159253 rows × 2 columns

Figure 14: Two new features added (Polarity & Subjectivity)

In addition, regarding ethical issues, whether users know that their comments are collected and analyzed, whether data use is licensed and authorized, and whether the generated model can be used for commercial purposes, all of these behaviors may violate the rights and interests of data producers and collectors. As machine learning penetrates into data analysis of all walks of life, the violation of personal privacy and rights and interests is likely to cause concerns from all walks of life. More laws need to be introduced to further define where the boundary between privacy protection and data analysis is and how to make appropriate trade-offs and trade-offs.

6 Conclusion

By using different classifiers and model optimization to predict sentiment classification of Twitter comments in the given datasets, this paper considers that the logistic regression model is optimal because after processing with bagging method, the accuracy of this model in developing set reaches 67%, which is more stable and satisfactory compared with other models.

In addition, data selection, user discrimination and possible bias and ethical issues in the analysis process are also worth paying attention to. This report can introduce more new features into future analyses, such as polarity and subjectivity of emotions, to further improve the accuracy of our predictions.

7 Bibliography

1. Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12), 2009.
2. Vadicamo, L., Carrara, F., Cimino, A., Cresci, S., Dell'Orletta, F., Falchi, F., & Tesconi, M. (2017). Cross-media learning for image sentiment analysis in the wild. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 308-317).
3. Tripathy, A., Anand, A., & Rath, S. K. (2017). Document-level sentiment classification using hybrid machine learning approach. Knowledge and Information Systems, 53(3), 805-831.
4. Liu, Y., Bi, J. W., & Fan, Z. P. (2017). Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms. Expert

Systems with Applications, 80, 323-339.

5. Twitter dataset (Eisenstein et al., 2010)
6. Blend datasets (Deri and Knight, 2015; Das and Ghosh, 2017; Cook and Stevenson, 2010)