COMP90049 Assignment 2 Report

Anonymous

1. Introduction

The project aims at applying several machine learning methods to classify which emoji the content should belongs to and make prediction for given datasets and evaluate the performance of machine learning methods in this scope.

1.1 Dataset

We take two kinds of data in our project: the raw twitter data, contains original text without emojis, the most 100 data, contains top 100 frequency tokens appears in raw data and the top 10 data which we will not use.

For each of the three kinds data above, we got three files. First is the train file, which contains contents and labels, we use it to train and achieve a model with a certain method, and use the model to make predictions. Then the dev file, which contains contents and labels different from train data. We apply our model on this file and make predictions and compare our predictions with the correct labels provided by this dev data in order to conclude the performance of the machine learning method we apply. Finally, the test file, which contains only the contents while its labels remain unknown. We are required to predict the label for each content, namely, classify each content as a certain emoji.

1.2 Implementation

In this project, we will apply K Nearest Neighbours (knn) [1], Support Vector Machines (svm) [2] and Tree Methods [3][4] as our classification methods to establish models and operate on test data to predict labels. Finally, the paper will discuss and compare performance of different machine learning methods

The project will be implemented with python, specifically scikit-learn [5], an open source machine learning library for Python.

1.3 Related Artefacts

There is the paper concern about Turkish movie reviews [6]. The author applies python NLTK (natural language tookit) package to make sentiment analysis on whether the movie reviews are positive or negative. The article follows training, testing, evaluating and classifying. It mainly applies Naïve Bayes as well as SVM and their derivation methods such as Multinomial Bayes and linear SVC. The project concerns more on raw data processing, that is, how to gather the appropriate information we require. And the result is quite acceptable, the linear svc method works out 100% accuracy.

2. Evaluation Metrics

In this project, we introduce sklearn. metrics.classification_report to provide a formal classification metrics. When it comes to multi-class situation, macro-average is used to do average on each class and then average for all classes

```
report = metrics.classification_report(target, predict)
```

This method calls for the target data, which is the fact, and the predicted data to achieve three kind of features we may use in this paper.

- Precision: shows percentage of right predictions among amount of predictions made for that class.
- Recall: is ratio of right predictions of a certain class to amount of that class in fact.
- F1_Score is introduced to balance the precision and recall, it is the harmonic mean of precision and recall in definition.

3. Methods and Evaluation

3.1 Tree Methods

We first apply random forest which is a clear and easy method. Compare to the decision tree method, random forest works better in data sets with large number of dimensions.

One of the very enjoyable feature of forest is that it can make more acceptable prediction within less time. As what we get:

```
ubuntu@kt:~$ python analyze.py train_most100.csv dev_most100.csv forest
forest model establishment time: 2.78662586212s
forest model prediction time: 0.449461936951s
             precision    recall  f1-score   support

       Clap       0.61      0.55      0.58      1265
        Cry       0.54      0.51      0.52      1587
 Disappoint       0.41      0.22      0.29       461
    Explode       0.48      0.58      0.52      1334
   FacePalm       0.47      0.33      0.38       433
      Hands       0.82      0.69      0.75      1322
    Neutral       0.31      0.47      0.37      1496
      Shrug       0.37      0.34      0.35      1222
      Think       0.55      0.46      0.50      1420
     Upside       0.38      0.39      0.39      1624

avg / total       0.50      0.48      0.48     12164
```

Figure 3.1

It only takes 2.78 seconds to learn and

establish the model and 0.45 seconds to predict, which is much quicker than any other methods we mentioned in this paper. What's more, it gains the highest precision rate at 50% among all methods, that is to say, the random forest works out the best result within least time, it is the most efficient method in this paper.

We take from dev_most100 the eleventh to fifteenth twitter data as an example:

	211	212	213	214	215
	at	day	for	and	be
	happy	got	i	i	if
	last	me	it	me	in
	of	the	my	need	it
	rt	today	now	people	this
	so		rt		
	you				
	your				
target	Hands	Upside	Explode	Upside	Upside
forest	Hands	Upside	Explode	Neutral	Neutral

Figure 3.2

The figure 3.2 shows what high-frequency tokens the twitter contains, what its origin emojis is and the emojis the random forest classifier gives us. As 214, "I, me, and, need, people" actually calls upside while the forest gives neutral. Another incorrect point happens in 215, as "be, if, in, it, this" should come up with upside but comes neutral. This is actually nothing with the method because these words can appear in any themes.

3.3 Support Vector Machine

SVM is to find a border to separate two classes. Namely, svm is created initially for two-class problems. We use svc which is an one-against-one method which requires n*(n-1)/2 classifiers to be established.

```
ubuntu@kt:~$ svm model establishment time: 710.750686169s
svm model prediction time: 114.317839861s
             precision    recall  f1-score   support

       Clap       0.33      0.54      0.41      1265
        Cry       0.39      0.38      0.38      1587
 Disappoint       1.00      0.10      0.17       461
    Explode       0.76      0.31      0.43      1334
   FacePalm       0.46      0.11      0.17       433
      Hands       0.78      0.57      0.66      1322
    Neutral       0.23      0.38      0.29      1496
      Shrug       0.34      0.12      0.18      1222
      Think       0.48      0.36      0.41      1420
     Upside       0.26      0.47      0.33      1624

avg / total       0.46      0.37      0.37     12164
```

Figure 3.3

The svm is a method that actually cost much when learning, especially in this case, several svms required to solve the multi-label situation. It finally takes nearly twelve minutes to learn and almost 2 minutes to predict. SVM also cost much space. It takes about 80M in the buff/cache.

```
ubuntu@kt:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3.9G         42M        3.7G         17M        126M        3.6G
Swap:            0B          0B          0B
```

Figure 3.4

About 46% predictions are correct with svm, but f1_score is the lowest among three methods.

	211	212	213	214	215
	at	day	for	and	be
	happy	got	i	i	if
	last	me	it	me	in
	of	the	my	need	it
	rt	today	now	people	this
	so		rt		
	you				
	your				
target	Hands	Upside	Explode	Upside	Upside
svm	Hands	Upside	Cry	Upside	Explode

Figure 3.5

Seems more information given, more possibility a correct classification is made.

3.4 K Nearest Neighbors

We apply 3 nearest neighbors to compare with other methods and do some other knn algorithms to compare with 3-nn itself.

```
ubuntu@kt:~$ python analyze.py train_most100.csv dev_most100.csv knn
knn model establishment time: 4.99710297585s
knn model prediction time: 101.157677889s
             precision    recall  f1-score   support

       Clap       0.34      0.67      0.45      1265
        Cry       0.41      0.53      0.46      1587
 Disappoint       0.19      0.26      0.22       461
    Explode       0.49      0.49      0.49      1334
   FacePalm       0.34      0.31      0.33       433
      Hands       0.55      0.72      0.62      1322
    Neutral       0.46      0.33      0.39      1496
      Shrug       0.51      0.27      0.35      1222
      Think       0.71      0.38      0.50      1420
     Upside       0.46      0.29      0.36      1624

avg / total       0.47      0.44      0.44     12164
```

Figure 3.6

3NN is a simple method without learning involved, which takes only 5 seconds to build up a model but 101 seconds to predict. And it gains 47% precision with f1_score 0.44, a quite acceptable performance.

KNN in classification is to assign then label depends on the object's K nearest neighbors. That leads to the fact that the larger K is, the less variant the prediction will be.

We applied 3 nearest neighbors as our normal gains and do 2, 5 and 7 nearest neighbors to observe the performance of different K parameter and its influence.

	precision	recall	f1-score
2-nn	48%	48%	53%
3-nn	47%	44%	45%
5-nn	46%	45%	44%
7-nn	44%	44%	44%

Figure 3.6

In this case, since each twitter data is independent, while some of them maybe related with each other on a certain topic, we will gain a higher precision because we won't be interfered by that many other objects.

	22	23	24	25	26
	all	for	all	it	has
	re	up	get		have
	the	this	rt		in
	when	you	this		people
	time		today		the
			well		this
			who		up
			you		you
target	Upside	Upside	Clap	Think	Explode
knn-2	Upside	Hands	Clap	Disappoint	Explode
knn-7	Cry	Hands	Clap	Neutral	Explode

Figure 3.7

Compare 2 with 7 nearest neighbors, 2nn works a bit better than 7 in that 2nn receives less impact from its neighbors. As what is shown in 22, the word all, the, time may be used in high frequency so that lead to less distance with objects contain these terms.

4. Conclusion

We apply three methods in classification, nearly 50% result are correct and random forest list top one in both time cost, only about 4 seconds in total, and precision, while svm, cost much time, about 15 minutes without the best result, seems not quite appropriate in this case. Knn is mediate, but also cost much time (about 2 minutes).

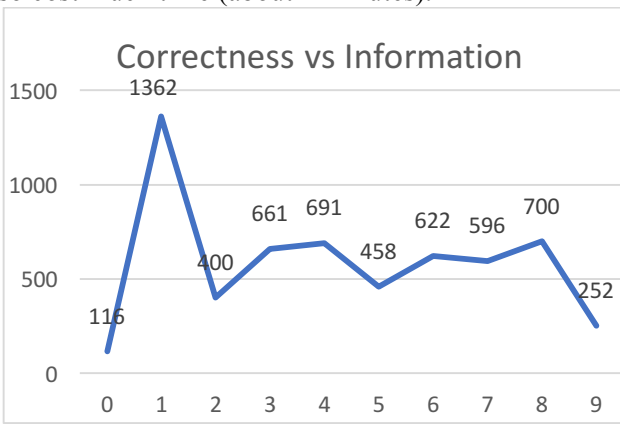

Figure 4.1 (x: amount of token in one data, y: amount of correct classification)

We also use python to count how many correct classifications are made against amount of information provided. As figure 4.1, we take random forest as an example, the twitter data that provide only one valid tokens is classified successfully 1362 times while more information, 9 tokens provided, only leads to 252 times correct prediction. It seems more token fail to gain improvement in the classification step.

The data we use to train, test and classify is the tokens with top 100 highest frequency. However, this lead to the fact that our data contains much meaningless words like "in, up" the preposition, "am is are" the be verb or "when, what" the conjunction. Therefore, although two sentences account for different meanings, namely different emojis, the key words may be omitted due to the top 100 frequency list. Under this situation, the two sentences may be classified into one class, which is the wrong prediction.

One potential solution is to reconstruct the most_100 data, generating a list of words without the words only required to form sentence like "is". Use more characteristic tokens like "happy, sad".

In future enhancement, we may consider applying unsupervised methods to setup cluster for similar twitters.

References

[1] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879

[2] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273-297. doi:10.1007/BF00994018

[3] Kamiński, B.; Jakubczyk, M.; Szufel, P. (2017). "A framework for sensitivity analysis of decision trees". Central European Journal of Operations Research. doi:10.1007/s10100-017-0479-6

[4] Ho, Tin Kam (1995). Random Decision Forests(PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[5] http://scikit-learn.org/stable/

[6] Performance analysis of supervised machine learning techniques for sentiment analysis. 2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS), Sensing, Signal Processing and Security (ICSSS), 2017 Third International Conference on. 128, 2017. ISSN: 978-1-5090-4929-5.