COMP30027 Report

Anonymous

1 Introduction

Nowadays, social networks such as Twitter has become an important component in our daily lives. The concept of "geotagging" is deeply investigated among those organizations who are interested in analyzing their user data. The aim of this project is to develop a classifier which can indicate the location based on the tweet text. Any feature engineering methods and classifiers that are used in this project is introduced.

The research mainly focused on natural language processing and text classification. Deep appreciation for George (2017) and Sebastian (2014)'s contribution.

2 Feature Engineering

The raw data in this project are tweet texts. Natural language processing technique is considered in order to generate useful data representation.

2.1 Data Pre-processing and Feature Extraction

In this project, three data pre-processing techniques are used and co-operate with different classifiers to find the combination that gives the best performance.

2.1.1 Count of High Frequency Words

The first data pre-processing option is to use the given top file, which takes a list of the high frequency words as its features. The feature words also indicate high mutual information. This data representation aims to demonstrate whether those words presence or absence for a particular instance. This method is considered to be inefficient since it did not explain features such as unique words and word combination from raw data.

2.1.2 Count Vectorize

Count vectors forms a corpus that consist of a range of unique tokens from the data. It then forms a data matrix which use the tokens as the attributes and record the token frequencies for each instance. This method is better than counting top high frequency words since it does not drop to much data feature but makes each instance more identical to each other.

2.1.3 TF-IDF[1]

The first component term frequency refers to 'Number of times term t appears in an instance/Number of terms in the instance'. The second component inverse document frequency is calculated by formula 'log(Total number of instances/ Number of instance with term t in it)'. The scores for each term are put into a representing matrix. The good of this method is that it reduced the weighting of common words among instances such as "is" and "the" but highlight the words that are identical to each instance. Word level gives score of each term. N-grams[2] and character level[3] looks into the combination of words and characters.

2.2 Feature Selection

χ^2 is used as the score function to apply feature selection. In this project, we assume that tweets in each area is identical due to reasons such as the use of language or particular vocabularies. Count vector and tf-idf are methods that attempt to extract as much information from raw data. It is not guaranteed that feature selection can drop irrelevant attributes during natural language process. Hence, the accuracy of datasets without feature selection are also determined and we take the one with higher accuracy as the result.

3 Class Evaluation

In this section, the performance of a range of classifiers over development dataset is determined. The confusion matrix of the prediction

[1] Term Frequency-Inverse Document Frequency

[2] default 2 3 grams, usually result in over-fitting and time consuming

[3] default 2 3 grams

with the highest accuracy for that classifier is analyzed.

3.1 Naive Bayes

Navie bayes classifier is a probabilistic classifier based on Bayes theorem. Multinomial NB[4] classifier is one of the most used classifiers for document classification. As Table 1 demonstrated, it shows steady but high performance. Based on Table 2, the confusion matrix demonstrates systematic errors. The errors are evenly distributed throughout each classes. This fact may due to the conditional independence assumption of the classifier. The deceptive conditional probability causes the errors. However, NB classifier is still a competitive choice that provide valuable and fast prediction.

Input data	Accuracy
Top100	0.30246
Count Vector	0.34098
Word Level TF-IDF	0.34120
Ngrams Level TF-IDF	0.29853
Character Level TF-IDF	0.34021

Table 1: Accuracy Tale for NB Classifier

Act/Pre	Bri	Per	Mel	Syd
Bri	4027	1900	1811	1591
Per	2765	3082	1881	1601
Mel	2677	1964	2934	1754
Syd	2859	1884	1825	2761

Table 2: Confusion Matrix for NB Classifier

3.2 Logistic Regression

Logistic regression is a regression method which is able to classifier and handle categorical data. It gives better performance if there is less unrelated attributes. Hence, its performance sometimes relies on good feature engineering. The good of this classifier is that the process is highly interpretable and easy to regularize. It also outputs predicted probabilities for instances. As Table 3 shows, logistic regression performs good with character level tf-idf. Its errors on the confusion matrix indicates that it may suffers from non-linearity and complete separations. To sum up, logistic regression may not be the best option for text classification, however, still a good choice as a baseline learner.

[4]Navie Baye

Input data	Accuracy
Top100	0.29873
Count Vector	0.33427
Word Level TF-IDF	0.32091
Ngrams Level TF-IDF	0.30031
Character Level TF-IDF	0.33849

Table 3: Accuracy Tale for Logistic Regression Classifier

Act/Pre	Bri	Per	Mel	Syd
Bri	3202	1602	2068	2457
Per	2163	2659	2135	2372
Mel	2051	1562	3180	2536
Syd	2172	1569	2114	3474

Table 4: Confusion Matrix for Logistic Regression Classifier

3.3 Random Forest

Random forest is an ensemble learning method based on the concept of decision tree and bagging[5]. In short, it randomly generate a range of decision trees over the data, then merge them to evaluate a less variance classifier that provides both classification and regression. The beauty of this method is that it effectively reduced overfitting issue. Furthermore, it make good prediction(see Table 5) even under the default hyperparameter. Nevertheless, it is difficult for us to do error analysis for random forest. Due to its randomness, the behaviour and process is extremely hard to describe and interpret. In sum, although random forest is usually time consuming and unable to interpret, it is still provides high quality predictions and worth an attempt.

Input data	Accuracy
Top100	0.27601
Count Vector	0.33918
Word Level TF-IDF	0.33211
Ngrams Level TF-IDF	0.28201
Character Level TF-IDF	0.32890

Table 5: Accuracy Tale for Random Forest Classifier

3.4 Support Vector Machine

Linear support vector machine(SVM) is a non-probabilistic classifier that works by fitting in-

[5]Bootstrap Aggregating, discussed later in the text.

Act/Pre	Bri	Per	Mel	Syd
Bri	3380	1659	2323	1967
Per	2396	2446	2404	2083
Mel	2582	1662	3069	2016
Syd	2591	1682	2272	2784

Table 6: Confusion Matrix for Random Forest Classifier

stances to its suitable category. Its kernel trick makes SVM powerful handling complex issues and data. It is a reasonable choice when we have limit understandings and knowledge towards the data even through it is usually time consuming with large dataset. It can also avoid over-fitting by tuning hyper-parameter. Table 5 indicates that SVM provides predictions with fair quality under default hyper-parameter. The biggest disadvantage of this classifier is the difficulty of tuning the hyper-parameters. SVM is hard to visualize and interpret the final model, which makes adjusting the hyper-parameters even harder. Overall, SVM still gives impressive performance if a fine hyper-parameter and a suitable kernel function is chose.

Input data	Accuracy
Top100	0.27933
Count Vector	0.33300
Word Level TF-IDF	0.34017
Ngrams Level TF-IDF	0.28342
Character Level TF-IDF	0.33721

Table 7: Accuracy Tale for SVM Classifier

Act/Pre	Bri	Per	Mel	Syd
Bri	3115	1696	2117	2401
Per	1983	2806	2162	2378
Mel	1849	1646	3354	2480
Syd	2038	1658	2190	3443

Table 8: Confusion Matrix for SVM Classifier

4 Ensemble Method

Ensemble method apply multiple weak or baseline learners to train and use them to build a better learner. In this project, bagging, boosting and stacking used to improve the prediction accuracy.

4.1 Bootstrap Aggregating

Bootstrap aggregating, also known as the bagging method, train the baseline learners by randomly assigning subset of the data these classifiers. Each baseline classifier has equal weight on voting. This method effectively reduce the variance as well as improve the accuracy. The most common use of this method is decision tree. As discussed above, random forest is indeed a bagging application on the decision trees. As Table 9 shows, bagging results in a significant improve for most of the classifiers. Among all the classifiers, NB classifier with word level tf-idf shows impressive predict accuracy.

Classifier Setup	Accuracy
Count Vector + NB	0.37821
Count Vector + SVM	0.35311
Count Vector + LR[6]	0.37512
Word Level TF-IDF + NB	0.38281
Word Level TF-IDF + SVM	0.36214
Word Level TF-IDF + LR	0.37224
Ngrams Level TF-IDF + NB	0.34214
Ngrams Level TF-IDF + SVM	0.30312
Ngrams Level TF-IDF + LR	0.33927
Character Level TF-IDF + NB	0.37982
Character Level TF-IDF + SVM	0.36134
Character Level TF-IDF + LR	0.37424

Table 9: Accuracy Tale for NB Classifier

4.2 Boosting

Boosting start with a baseline learner. Each turn it increase the weighting of the instances that are incorrectly predicted so that those "outstanding" instance gains more attention in the next turn. The aim of this method is to reduce the bias in order to gain higher accuracy. The most common application for boosting is Ada-boosting, which is accepted in this project. As Table 10 shows, boosting has a positive impact on the accuracy. Since boosting attempt to fit the given data with the lowest bias, however, it usually cause the problem of over-fitting.

4.3 Stacking

Stacking uses a meta classifier to combine the predictions of the weak classifiers. The good of this technique is that it can collaborate with different models to deal with complex data. In this project, we bootstrap aggregated or boosting all

[6]Logistic Regression

Classifier Setup	Accuracy
Count Vector + NB	0.37608
Count Vector + LR	0.37520
Word Level TF-IDF + NB	0.38042
Word Level TF-IDF + LR	0.37941
Character Level TF-IDF + NB	0.37021
Character Level TF-IDF + LR	0.36150

Table 10: Accuracy Tale for NB Classifier

the classifier used in stacking method and gain a remarkable effort.

Meta Classifier	Baseline Classifiers	Accuracy
NB	NB+LR+RF	0.30907
LR	NB+LR+RF	0.30032
bag(NB)	bag(NB)+ bag(LR)+bag(RF[7])	0.38941
boosting(NB)	boosting(NB)+ boosting(LR)+ booating(RF)	0.37240
bag(LR)	bag(NB)+ bag(LR)+bag(RF)	0.36621
boosting(LR)	boosting(NB)+ boosting(LR)+ booating(RF)	0.37998

Table 11: Accuracy Table based on Word Level TF-IDF

5 Conclusions

Among all the classifier combinations we discussed above, a stacking classifier using bagging multinomial Naive Bayes classifier as the meta classifier on word level tf-idf data gives the best performance. Multinomial Naive Bayes classifier gives fast but accurate prediction if its assumptions on conditional independence holds in real life. In this case, we indeed assumed that the tweets from each location differs to some extent such as some unique combinations of words and emojis or even keywords that directly related to some local features. As long as the real life circumstance satisfy the assumption of the NB classifier, classifier will guarantee powerful performance.

The best accuracy gain in this project is close to 0.4, which is considered to be a fairly high effort based on the given techniques. There are other attempts we may consider in the future. Blending method may help improve the accuracy even higher. A neutral network combined with word2vec also worth an attempt.

Word Count:1355

[7]Random Forest

References

Trevor Cohn Afshin, Rahimi and Timothy Baldwin. 2015. Twitter user geolocation using a unified text and network prediction model.

elitedatascience. 2017. Dimensionality reduction algorithms: Strengths and weaknesses.

K. George. 2017. Geotagging text content with language models and feature mining.

MonkeyLearn. 2017. Text classification a comprehensive guide to classifying text with machine learning.

Donges Niklas. 2018. The random forest algorithm.

Nss. 2017. An intuitive understanding of word embeddings: From count vectors to word2vec.

Gandhi Rohith. 2018. Naive bayes classifier.

Ongaya Sammy. 2018. Feature extraction with tf-idf.

Raschka Sebastian. 2014. Naive bayes and text classification introduction and theory.

BANSAL SHIVAM. 2018. A comprehensive guide to understand and implement text classification in python.

Statinfer. 2016. 204.6.8 svm : Advantages disadvantages and applications.