Geolocation classifier of Tweets

Anonymous

1 Introduction

Location information of a social network user has plenty of applications. With increasing population there is surplus amount of data to perform data analysis and various Natural Language Processing Tasks.

In this paper, the use of supervised machine learning approaches are explored by taking into account linguistic features of words and also some dialect elements of the user location to determine the location of a tweet. For this task, the Twitter dataset from Eisenstein et al. (2010) and Rahimi et al. (2018) was taken. The dataset consists of over 96,000 tweets that are tagged with one of New York, Georgia or California as their location.

The raw tweets are taken and various preprocessing techniques are applied. Following which features are generated based on dialects and linguistic features. Then we use the bag of words notation to represent the tweets. The results are obtained using Linear Support Vector Machine(SVM) based classification and Naive Bayes Classification. The obtained results are analysed and studied in this paper.

2 Related work

Predicting location based on social network text has been attempted in the past. Chi et al. (2016) have also attempted to predict the location of tweet as a classification task. They have used publicly available Geoname dataset. They then group cities and formulate Location Indicative Words(LIW) as features. This research paper highlights the importance of choosing good features.

Thomas and Hennig (2018) have performed this task without the use of any external knowledge sources or corpora. They have made use of Neural Networks that is trained only on the twitter text and meta data.

On the other hand there are attempts to address this task using semi supervised approaches as seen in Cha et al. (2015) and Rahimi et al. (2018). Cha et al. (2015) use sparse coding with the help of Knowledge sources and Rahimi et al. (2018) proposed a sophisticated model that uses Graph Convolution Networks. These approaches showcase semi-supervised approaches to tackling this problem.

3 Methodology

In this implementation, the Support Vector Machines(SVM) are used as a classifier to classify the tweets. For a comparative study, same experiments were also conducted using the Naive Bayes Classifier. Figure 1 shows the overview of the system.

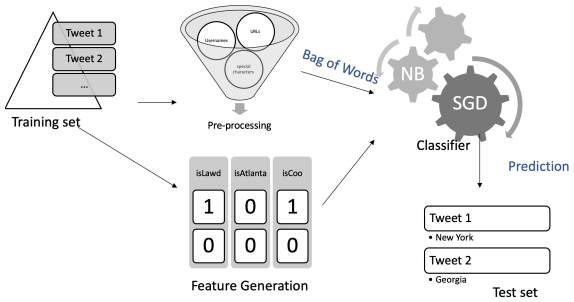

Figure 1: Overview of System

3.1 Pre-Processing of raw data

The dataset used here has raw tweets with special characters, URLs, mentions of other twitter usernames and numbers. This data does not add much value during prediction of our location classes. Hence it is important to cleanse this data before classification. In this step, several processing is performed to raw tweets using available inbuilt libraries[1]. The models used by the feature generation and classification modules are trained on these pre-processed datasets. This makes them more robust to variances in

[1]Natural Language Took Kit in Python was used for preprocessing

patterns and term frequencies. The tweets are then tokenized using NLTKs tokenizer and then stemming is performed which helps in building stronger feature sets as we extract the root of many words in the tweets.

3.2 Feature Engineering

Machine learning techniques mainly rely on features of the text. Choosing the right features improve the quality of the machine learned heuristic. However, in theory there are innumerable amount of potential features to pick from. In this implementation, a number of features were considered. The number of questions, the number of hashtags, number of ellipses, number of exclamations, number of mentions and number of URLs were taken which fall mainly into the linguistic features category. Figure 2, 3 and 4 [2] shows the how the tweets get separated after introducing these features. Additionally, some frequently used words (see Table 1) were also used as features which are the dialect features. These were taken from online sources and some also through analysis of training tweets dataset.

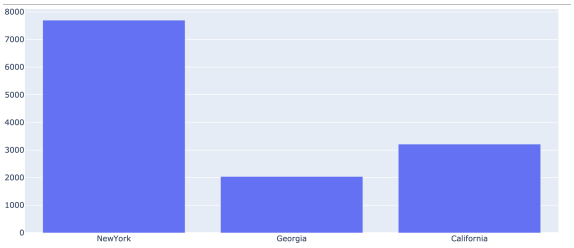

Figure 2: Tweets separation with number of questions feature

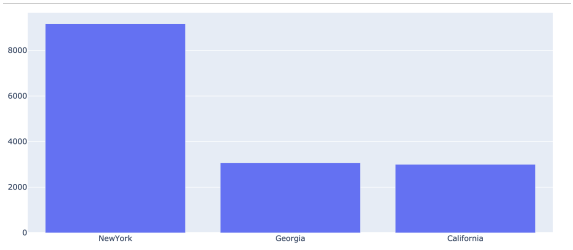

Figure 3: Tweets separation with number of Hashtags feature

[2] All graphs are plotted using sklearn and plotly python library

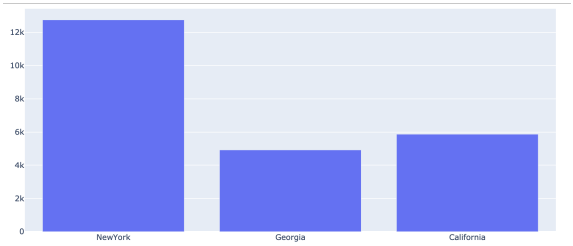

Figure 4: Tweets separation with number of exclamation feature

NewYork	California	Georgia
Deadass	Hella	atlanta
sus	Yee	georgia
brick	Bomb	Yall
brolic	socal	Lawd
odee	nocal	sir
wack	Coo	

Table 1: Common words used as features for each state

3.3 Bag of words conversion

Before building a Bag of words model a number of steps are followed. A wordlist is built by counting the number of occurrences of every unique word across all of the training dataset. The most commont words are the typical english stopwords. These are filtered out from wordlist using nltk. The most common words in the dataset after stop words removal is given in Table 1 Before building the final bag of words, the words occurring too frequently were not considered and after analysis the word count between 10 and 10000 were taken to build the bag of words representation of the tweets. The most common words in each class is show in Figure 5

Word	Count
rt	20650
u	15725
lol	15191
im	10362
like	5660

Table 2: Most common words in Training set

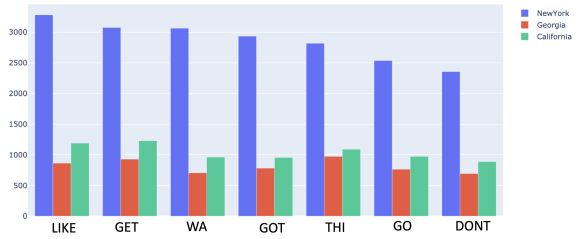

Figure 5: Most common words in each location

3.4 Classification using Machine Learning

3.4.1 Naive Bayes Classification

The Naive Bayes classifier is a probabilistic classifier that is mainly based on the Bayes Theorem. In this implementation, the Bernoulli Naive Bayes classifier is studied. This takes only boolean values as features. In our tweet representation using Bag of Words, and using 'if dialect word' is present feature, we have a boolean representation of our tweets hence the Bernoulli Naïve Bayes Classifier[3] is used.

The main advantage of this implementation is that it is fast and also easy to implement. However, the disadvantage is that it considers the feature to be independent which may not be the case. For example, 'lawd' word which is a feature of Georgia class may be used in tweet only when there is a question mark. This gave a comparable accuracy of around 2% lesser than the SVM based classifier.

3.4.2 Support Vector Machine(SVM) based Classification

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide and returns a best fitting hyperplane that divides, or categorizes the data. After getting the hyperplane, some features are fed to the classifier and the prediction is obtained. In this implementation, sklearns's SDGClassifier was used which is a Linear classifiers (SVM) with Stochastic Gradient Descent(SDG) training.

This classifier works well with data represented as dense or sparse arrays of values for the features. It is effective in high dimensional space. In our implementation we have sparse arrays as features due to the bag of words notation and also the common words feature. This case is handled well by the classifier ansd is the main motivation for choosinh this method. This takes more time to train than the Naive Bayes Classifier but gives more accuracy.

[3]All classifiers are taken from pythons sklearn library

4 Evaluation

In this implementation we evaluate the performance in classifying each tweet to its corresponding location which can be either of NewYork, Georgia or California. The test dataset is taken and its predictions are analysed. For evaluation of this system, the F1 score and Accuracy are the best metric to show the performance. This is mainly because this is a classification problem with three classes and with the help of F1 score this can be accurately depicted for each class. The table below has the consolidated report of the results obtained using the SDG Classifier (see Table 3).

	NewYork	Georgia	California
F1	0.122	0.052	0.77
Precision	0.570	0.459	0.64
Recall	0.068	0.028	0.98
Learing time		11.645s	
Predicting time		0.3986s	
Accuracy			0.64114

Table 3: Evaluation Report of the SGDClassifier

The classifier has an accuracy of 64.114%. The same predictions were done using the Naive Bayes classifier. This achieved an accuracy of 63.06% (see Table 4). In both classifiers it can be seen that the model is very good at predicting the California class as it has both high precision and recall, followed by NewYork and then Georgia.

	NewYork	Georgia	California
F1	0.227	0.146	0.772
Precision	0.412	0.343	0.664
Recall	0.156	0.092	0.921
Learing time		1.3320s	
Predicting time		0.956s	
Accuracy			0.6306

Table 4: Evaluation Report of the Naive Bayes Classifier

5 Critical Analysis

A number of analysis and inferences can be made on this implementation of geolocation classifier.

1. Both the classifiers give us similar accuracy with SVM based SGD classifier performing slightly better than Naïve Bayes. However, from table we can see that there is difference in the learning time. The learning time in case of Naïve bayes is only 1.5 seconds whereas SGD takes 11.6 seconds. This is more than ten times the Naïve bayes value. This is mainly because of the simplicity of the Naïve bayes classifier.

2. Feature Engineering is the crucial part of a classification algorithm. In this implementation, many features were considered. But after repeated analysis it was found that some features are not as important as the others and removing these features showed an improvement in the accuracy. For example, from Figure 3 we can see that the hashtags feature does prove to be a useful heuristic for the classifier as the hashtags are used by all classes. Removing this showed an improvement of 0.2% in the accuracy.

3. The wordlist is the basis for the bag of words(BOW) representation. It becomes important to choose the right words to represent the tweet. For initial analysis, the words which have occurred at least 100 times and at most 10,000 times we taken. Upon changing this range to 10-10,000, There was an improvement in the accuracy of 0.5%. This means to say that the fewer occurring words played an important role in the prediction.

4. This implementation was performed by exploring many features before settling with the linguistic features (number of questions, URLs, mentions) and dialect features (factual common words used in those classes). Table 4 shows comparison of classification performance when we considered only linguistic and when both were taken. Clearly adding the common class words feature improved the accuracy.

Accuracy%	linguistics features	adding dialect features
SGD Classifier	64.008	64.114
Naive Bayes Classifier	62.98	63.06

Table 5: Effect of features on both Classifiers

6 Conclusion

This system achieves an accuracy of 64.14% by considering linguistic and dialect features. By performing repeated analysis, it was found that features considered play an important role in the prediction of classes. With the help of additional features and other deep learning techniques this accuracy can be improved.

References

M. Cha, Y. Gwon, and H. Kung. Twitter geolocation and regional classification via sparse coding, 2015.

L. Chi, K. H. Lim, N. Alam, and C. J. Butler. Geolocation prediction in twitter using location indicative words and textual features. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 227–234, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee.

J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1277–1287, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

A. Rahimi, T. Cohn, and T. Baldwin. Semi-supervised user geolocation via graph convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2009–2019, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1187.

P. Thomas and L. Hennig. *Twitter Geolocation Prediction Using Neural Networks*, pages 248–255. 01 2018. ISBN 978-3-319-73705-8. doi: 10.1007/978-3-319-73706-$5_2$1.