Classification of Tweets according to Geographic Location with Machine Learning

Abstract

With increasing growth of data in terms of both volume and variety, finding patterns and relationships to understand the source (geographic location) of public views and opinions became extremely critical for various purposes.

1 Introduction

Task was to classify each Tweet according to geographic location of where it was made. Separated training and development data were provided. Each instance had Tweet ID, User ID, Tweet Content and Location Class as attributes.

In this report, combination of SciKit Learn[1] and users Tweets from 2008 was used to determine which US city each tweet was being sent from. Zero-R and One-R were used as baseline measurements against Random Forest and Naïve Bays. The report will discuss the effectiveness of such classifiers and feature sets used.

2 Feature Engineering

Although optional, custom features were engineered.

2.1 Features

The following sections show the features used. A group of features from here were used to model the tweets.

2.1.1 Related Words Vector

Related words were extracted from Semantic Link[2] and formed a group of words (see Table 1). Every time a word in the group was encountered, it contributed a value of 1 to the resulting vector. For example, text 'boston astros san sd' would be $\langle 1, 1, 2, ... \rangle$.

This feature was used after the assumption that people different cities use set words that are different from the others. This feature models their words used.

Group	Related Words
Boston	boston, celtics, bruins, fenway, berklee, sox, roxbury, mbta ...
Houston	houston, astros, texans, oilers, galveston, tx, cougars, nutt ...
Seattle	diego, san, chargers, sd, sdut, sandiego, baja, obispo ...
⋮	⋮

Table 1: Example of related words according to the classes

2.1.2 Topic Words Vector

Selection of 'topics' were chosen manually and 5 synonyms were chosen as a group of words for that 'topic'. The synonyms were extracted from NLTK[3] and vector formulation worked exactly like Related Words Count.

This feature was used after the assumption that people different cities Tweet about specific group of common words that are different from the others. This feature models their talking topics.

2.1.3 Sentence Structure Vector

Tweet's sentence structure was examined using NLTK's tagger. A vector representation was built according to the tag. To allow position

[1]http://scikit-learn.org/ Machine Learning Libarary for Python

[2]http://semantic-link.com/

[3]http://www.nltk.org/

to be considered, when adding into the vector, the value added was a prime number for the position. For example,

"They refuse to permit us to obtain the refuse permit"

would be processed into

('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')

Then, if vector format was

$$\langle PRP, VBP, TO, NN...\rangle$$

the vector formation would be

$$\langle 2+11, 3, 5+13, 21+23, ...\rangle = \langle 13, 3, 18, 44, ...\rangle$$

This feature was used after the assumption that people different cities use specific patterned sentence structure. This feature models their sentence structure itself.

2.1.4 TF-IDF Similarity Vector

Using the training data provided, Tweets were arranged according to their classes. Then, similarity score was calculated using sum of TF-IDF Similarity score against the Tweets in the class. IDF is calculated based on all the Tweets made. Then, for each word in the Tweet, the frequency is multiplied by IDF to calculate it's similarity with all the previous Tweets in a city. Then, TF-IDF score is summed for final score for that city.

This feature was used after the assumption that training data set successfully represents all the possible Tweets that people normally send in a city. This feature models their similarity match with the Tweet history, therefore determining the city.

2.2 Feature Sets Used

Set A: *Related Words Vector, Topic Words Vector*
This was the 'obvious' feature set that was the starting point for adding features.

Set B: *Sentence Structure Vector*
This set was purely made to determine the performance of the feature. High performance of this feature would mean people use different sentence structure compared to other region.

Set C: *TF-IDF Similarity Vector*
This set was purely made to determine the performance of the feature. High performance of this feature would mean people write similarly within the region and differ from others.

Set D: *Related Words Vector, Topic Words Vector, Sentence Structure Vector*
This was set of relatively high performing features. This was, obviously, derived after previous sets.

Feature sets' performances were evaluated along with evaluation of classifiers' performances.

3 Classifier Evaluation Method

Holdout evaluation strategy was used, training the classifier with training data and evaluating the classifier with development data. This was used over cross-evaluation method since this allowed maximum learning with training data then evaluated on fresh, unlearned set to detect over-fitting.

Accuracy was calculated for each classifier against all sets. Accuracy was used over precision and F-Score since considering number of correct predictions over all would be a straightforward answer to the performance of the classifier.

4 Classifiers

Multiple Classifiers from SciKit Learn were put under test. SVM was not used due to its slowness in learning.

Set	0-R	1-R	NB	Forest
A	26.16%	26.16%	21.86%	30.42%
B	26.16%	26.16%	19.55%	26.16%
C	26.16%	19.23%	24.71%	24.00%
D	26.16%	26.16%	23.53%	30.37%

Table 2: Performance summary for each classifier against each feature sets.

4.1 Zero-R

Zero-R classifies based on majority class in the data set. It has almost no ability to predict however provides baseline for comparison for other classifiers, i.e. if the other classifier has lower accuracy than Zero-R, that classifier is ineffective and useless.

Because Zero-R does not care about the feature set, it maintained its accuracy of 26.16% (see Table 2) throughout the data sets.

4.2 One-R

One-R uses the best performing decision stump. It generates one rule for each attribute and selects the one with smallest error rate.

Although more advanced than Zero-R, it usually is too simple to accurately predict classes. As shown in Table 2, it performed almost identically as Zero-R except in Set C where it fell behind by about 5%.

The reason for falling behind in Set C is relatively straight forward. One-R considers only one of the features in mind. But features in Set C - being the TF-IDF similarity score vector - was designed to be used together because similarity scores mean little by themselves and holds meaning when relatively compared together. For example,

$$sim(Tweet, SE) = \langle 4.24 \rangle$$

holds little meaning towards similarity to SE. However,

$$sim(Tweet, SE, SD) = \langle 4.24, 1.42 \rangle$$

indicates that Tweet is more similar to SE compared to SD.

Reasons behind exact same performance with Zero-R for other sets is not trivial. The most likely situation is that One-R is constructing a decision stump for a specific attribute which results in choosing majority class all the time. Further investigation needs to be carried out to find details.

4.3 Naïve Bayes

Naïve Bayes assumes that distribution of classes in test set is the same as the training set and each features are independent from the other. This makes variety of training set and minimizing unseen data extremely important.

Naïve Bayes' common problem of over-fitting is present (see Table 3), where it performs better with the set it was trained with but less so with other sets. It's poor performance is likely due to the fact that NB performs poorly on unseen data. This could be due to features failing to capture generality of the Tweet contents or training data was inadequate and a lot of unseen data was remained.

The best performance was shown with Set C. This is expected since only Set C has true continuous attributes - which is a required assumption for Naïve Bayes to work.

Set	**Train**	**Development**
A	22.61%	21.86%
B	22.09%	19.55%
C	34.94%	24.71%
D	26.44%	23.53%

Table 3: Accuracy of NB on training and development data.

4.4 Random Forest

Random Forest creates multiple over-fitted Decision Trees using a subset of attributes. The prediction is decided by considering all the Decision Trees' results. Random Forest performed better than both baseline methods except on Set C (see Table 2). The higher performance is not surprising however reason for lower performance in Set C is not trivial.

Set	**Train**	**Development**
A	30.08%	30.42%
B	25.77%	26.16%
C	53.53%	24.00%
D	30.06%	30.37%

Table 4: Accuracy of Random Forest on training and development data.

Random Forest did not over-fit at all for Set A, B and D (see Table 4). It's drastic difference in accuracy for Set C can be due to many factors. The most likely is that Set C was not designed to cover general cases or training data did not represent all cases. Over-fitting is unlikely based on performance on other sets. This ultimately results in poor performance in general case and therefore performing poorer than baseline methods.

5 Performance of Classifiers and Feature Sets

Overall, All classifiers except Random Forest can be considered useless purely based on the accuracy on development data. Random Forest also managed to not over-fit to training data unlike Naïve Bayes. Therefore, Random Forest seems to be the best at classifying locations for the Tweets.

Set	Average Accuracy
A	26.15%
B	24.51%
C	21.03%
D	26.56%

Table 5: Average accuracy for each set.

Set C remains as a feature set with high potential. Not only Naïve Bayes performed the best with it but most critically, Random Forest seemed to learn the pattern extremely well (based on Table 4). This leaves us with some possible improvements to be made. One of them would be improving variety of training data so that similarity score represents more general cases for the class.

6 Other Classifiers and Potential Improvements

K-Nearest Neighbour is potentially ideal classifier. In real life, new Tweets are being made consistently. K-NN's ability to add extra data on the fly will help us drastically to learn latest pattern quickly.

If possible, learning from more training data would be ideal. Learning from both training and development data and doing M-Cross Validation will definitely result in better accuracy however that may not reflect our performance in real test data.

It may be possible that training data is biased for learning general patterns, making the classifier perform poorly (and over-fitting) no matter what model is used. However, this is hard to confirm without actually going through all the training data provided. In a glance, they were filled with duplicate Tweets and advertisements. These two problems are critically important to be dealt with since duplicate Tweets will inevitably lead to being biased and advertisements are mostly unrelated to advertiser's locations (For example, A store in the Seattle can advertise towards customers in Boston, using Boston-related words).

7 Conclusion

Usage of Zero-R and One-R provided a suitable baseline for the classifiers and accuracy gave us sufficient information regarding classifiers' performance. Although high-volume training data was ideal with quantity, its variety and bias are left to be verified.

Pre-processing and tampering the training data may be required to remove noise - duplicate and advertising Tweets. This is expected to improve performance of the classifiers overall.

It is unclear that the feature sets used to model the Tweets were reliable. It is hard to neither accept nor reject the reliability and generality of the models. Further investigations regarding this will be ideal.