Project 2 – Geolocation of Tweets

Introduction

The objective of this assignment was to classify each tweet in a set of tweets into one of five classes: Boston (B), Houston (H), San Diego (SD), Seattle (Se), and Washington DC (W). Various machine learning classification algorithms were utilized to assign a class to each tweet based on the contents of the tweet. This report evaluates the effectiveness of the three machine learning classification algorithms used to solve the problem: K-Nearest Neighbor, Naïve Bayes, and Decision Tree.

Methodology

A Python 3 software system was used to analyze the set of tweets and classify the test data into classes. The system implemented the scikit-learn machine learning library for Python to train various classifiers and classify the test data. The system took three input files: a training data set, a development data set, and a test data set. Each input is required to be in ARFF format. The system parsed the input and created models for each input data set where each tweet was represented by a vector space model for the best 35 features. The system then trained various classifiers with the training data, calculated evaluation metrics for the development data for each classifier, and provided predictions for the test data for each classifier. The results and analysis of each classifier are described in the subsequent sections.

K-Nearest Neighbors

The K-Nearest Neighbor classification algorithm was an obvious choice to solve this problem considering its simplicity and ability to handle multiple classes with ease. The system used both 1NN and 3NN to classify the test data. 1NN reported an accuracy of only 21.7%, which is likely because each tweet was classified according only to its nearest neighbor, which will provide a lower accuracy than considering the majority among several neighbors. However, 3NN only reported a 0.1% gain in accuracy over 1NN at 21.8%, which is likely due to the fact that most of the tweets did not contain any of the features. Thus, all of the tweets that did not contain any of the features were all placed into the same class even though in reality they were spread across several classes. This pattern also caused the class B to receive a recall of 97% while each of the other classes had a recall of less than 10%. This indicates that each tweet without any of the features was placed in class B, which makes sense because the nearest neighbor classification algorithm classifies each instance based on its neighbors. If many of the instances are identical, then those instances will all be placed in the same class because they are all nearest neighbors to each other. Although K-Nearest Neighbor algorithm is simplistic in concept, it was not effective in its ability to correctly classify the tweets due to its classification of identical instances into the same class.

Naïve Bayes

The Naïve Bayes classifier is known for its simplistic concept and high performance on various types of datasets. The underlying concept of the Naïve Bayes classifier is to classify an instance by using Bayes rule to calculate the probability that the instance was created by each of the possible classes and selecting the class with the highest probability. One of the most fundamental assumptions of a Naïve Bayes classifier is that every feature is independent, which is untrue in a real-world context but allows for the Naïve Bayes classifier's simplicity and speed even on large datasets [1]. Although Naïve Bayes classifiers are known to have much success in the field of document classification, the classifier used to solve this problem reported a 21.7% accuracy rate, which is the same accuracy rate given by 1NN. There seemed to be no improvement in using Naïve Bayes over K-Nearest Neighbor in solving the problem. Like with K-Nearest Neighbor, the Naïve Bayes classifier tended to classify identical instances into the same class, which in this case was W. This is unsurprising given that many of the instances were zero-vectors since many tweets did not contain any of the features. After training the classifier, every identical test instance would be labeled with the same class because the probabilities calculated for each of these identical instances would be the same. This explains why W got the majority of the identical instances as well as a recall of 99%. Since W contained the vast majority of all test instances, the recall was incredibly high because most correct W instances were placed in W. Conversely, every other class had a precision of 76% or higher, which indicates that the instances that were placed in classes other than W were correct most of the time because they contained enough features to calculate relevant probabilities for which class they should be in. Overall, the Naïve Bayes classifier had poor performance because of so many identical zero-vector test instances, but performed well on test instances that were nonzero vectors because it was able to calculate relevant probabilities and place the instance into the correct class most of the time.

Decision Tree

The Decision Tree classifier is very easy to conceptualize because the tree is constructed with nodes as tests on the attributes and branches as decisions based on the attributes. Each leaf represents a class that the instance should be placed into if they reach the leaf through the tree. There are many challenges with building a good Decision Tree classifier, because choosing the correct tree is considered to be NP-Hard due to needing to find the best way to split the training data based on local criterion. Additionally, trees are often over-split which can lead to overfitting [2]. Despite these challenges, Decision Tree classifiers are known to work well on both numeric and non-numeric data and work especially well when used in a Random Forest. The Decision Tree classifier yielded the highest proportion of true positives overall with an accuracy rate of 30.6%, which is almost 10% higher than any other classifier used to solve the problem. Unfortunately, this classifier had the same problem with identical zero-vector test instances as the other classifiers had, and tended to place these instances in the SD class. However, having manually looked through the predictions for the development data, this classifier had much more success in predicting the classes of the nonzero vector instances than the other classifiers. For example, Se had a precision of 90.6% which indicates that almost all of

the development instances that were classified as Se were correct. Overall, the Decision Tree classifier did the best job classifying the data because the tree structure allowed multiple routes to different classes based on tests on attributes, so there was much more freedom for each instance to find its way to the correct class based on its attributes.

Conclusions

All three classifiers had trouble correctly classifying tweets that did not contain any of the features because without features there is no way to distinguish between these identical instances that are in different classes in reality. This issue led to most test instances being placed into one particular class for each classifier. Overall, the Decision Tree classifier performed better than K-Nearest Neighbor and Naïve Bayes with an accuracy rate improvement of almost 10%. Decision Tree was better able to classify the tweets that contained features likely due to its tree structure and multiple routes to get to any given class. Naïve Bayes and K-Nearest Neighbor had similar outcomes because identical instances always led to identical results. The performance of any of these classifiers could be improved in regards to this dataset if more features were used to provide more of a distinction between the tweets and prevent the majority of the tweets being placed into the same class. Additionally, a larger training data set would decrease overfitting and allow the classifiers to generalize the data more effectively.

References

[1] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1998.

[2] Kumar, Vipin. Classification: Basic Concepts, Decision Trees, And Model Evaluation. 2016. Web. 12 Oct. 2016.