Twitter Relevance Filtering via Joint Bayes Classifiers from User Clustering

Alexander L. Churchill
achur@stanford.edu

Emmanouel G. Liodakis
liodakis@stanford.edu

Simon H. Ye
sye@stanford.edu

Dec. 12, 2010

1 Introduction

The task of classifying feed item data, such as email or RSS items, has been a subject of proposed learning algorithms since the mid 1990s. Unlike spam classifiers, relevance classification cannot use a universal training set, but rather must be trained by each individual user, however most relevance classifiers use an approach similar to spam classifiers. Most research in relevance classification has employed a classifier over an individualized training set, usually one of a naïve Bayesian classifier, support vector machines, or neural-network, case-based, or knowledge-based approaches [2] [3]. However, in practice, few of these algorithms have been implemented, and those that have enjoy only limited success. In large part, this is due to common user expectations with regard to classification accuracy. Users tolerate few errors and expect immediate results [1].

The past few years have seen a radical shift in the way users consume data. In particular, services like Twitter have dramatically increased social feed consumption. This poses the opportunity to build implicit social graphs to improve the quality of relevance classification over smaller training sets. In this paper, we examine the use of Hierarchical Clustering on Twitter users to build these implicit social graphs and to then use the multinomial Bayesian classification approach over the augmented training sets to do Tweet relevance classification. In this way, we leverage the power of social feeds to improve the quality and training speed of individualized relevance classifiers.

2 Dataset Description

The dataset consisted of approximately 1,200 tweets curated from 24 of the most influential users on twitter based on number of followers. This consisted mostly of celebrities, comedians, heads of state, and prominent news sources. The distribution was roughly even so approximately 50 tweets were gathered per tweeter. The user ratings were performed by 25 users, with each person rating approximately 600 tweets randomly sampled from half of the users in the dataset, totaling approximately 15,000 tweets. All ratings were performed in a binary manner for which the user either liked or disliked the tweet.

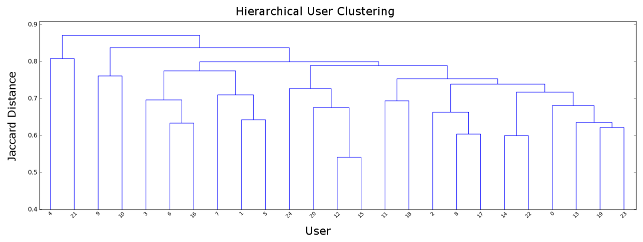
3 Algorithm Description

3.1 Data Preprocessing

The textual content for each tweet was tokenized through a multistage process. All urls were first parsed from the tweet and normalized to the base domain name of the linked-to site. Bit.ly links were also followed through and the resulting url was also stripped to the base domain name. To process the textual content, all punctuation was stripped from both sides of words while all casing was converted to lowercase. Tokenizing was performed using whitespace as a delimiter. Each token was then stemmed using the Porter stemming algorithm to conflate differently suffixed versions of the words into stemmed tokens. Finally for each tweet, a meta-token was added to indicate the author of the tweet. All of the tweets were then processed for the most significant bigrams according to chi-squared significance, of which the most significant 200 bigrams were also added to the token dictionary. Tweets were also processed similarly for training and testing phases of the classifier.

3.2 Hierarchical Clustering

Hierarchical Clustering progressively clusters data points by taking the two closest training examples and placing them as sibling nodes in a tree. These cluster centroids are then reweighted to represent consituent training examples before making the next clustering assignment. Twitter users are represented as sparse vectors in n-dimensional space, where n is the total number of tweets. Each element of the vector takes on a three values [-1, 0 ,1] where -1 corresponds to 'disliking' a tweet, 0 for no answer, and '1' for liking a tweet.



Users are clustered using the Jacard distance.

3.3 Multinomial Bayesian Classification using Clustering Results

Multinomial Bayesian Classifiers are a set of discriminative algorithms that predict the probability of classifying a tweet as either -1 or 1 based on its features by using Bayes' rule and calculating the predicate probability of each feature qualified on the rating of its tweet and the predicate that the feature exists in the tweet, which can be easily calculated with one pass through the data.

3.4 Mathematical Description of Algorithm

Given a training set:

$$\left((X^{(n)}, Y^{(n)}), ..., (X^{(n)}, Y^{(n)})\right)$$

where each $X^{(i)}$ is a vector of tweets and each $Y^{(i)}$ is a vector of their corresponding ratings $r \in \{-1, 1\}$, we let $W = \{w_1, ..., w_p\}$ be the set of all words in the training set.

For each $(X^{(i)}, Y^{(i)})$, define $\eta_j^{(i)} = \sum_{k=1}^n \sum_j 1_{\{X^{(i)}|Y^{(i)}=1\}}$ and define $\hat{\eta}_j^{(i)} = \sum_{k=1}^n \sum_j 1_{\{X^{(i)}|Y^{(i)}=-1\}}$. The multinomial Bayesian classifier applied to a single $(X^{(i)}, Y^{(i)})$ yeilds:

$$P(Y_k^{(i)} = 1|w_j \in X_k^{(i)}) = \frac{\eta_j^{(i)}}{\eta_j^{(i)} + \hat{\eta}_j^{(i)}} \text{ and } P(Y_k^{(i)} = -1|w_j \in X_k^{(i)}) = \frac{\hat{\eta}_j^{(i)}}{\eta_j^{(i)} + \hat{\eta}_j^{(i)}}$$

so we derive that

$$P(Y_\ell^{(i)} = 1|X_\ell^{(i)}) = \frac{\prod_{w_j \in X_k^{(i)}} P(w_j|Y_k^{(i)} = 1)P(Y^{(i)} = 1)}{\prod_{w_j \in X_k^{(i)}} P(w_j|Y_k^{(i)} = 1)P(Y^{(i)} = 1) + \prod_{w_j \in X_k^{(i)}} P(w_j|Y_k^{(i)} = -1)P(Y^{(i)} = -1)}.$$

Additionally, given our data set, we can calculate our sample expected value to estimate our expectation

$$E_\ell[P(Y_\ell^{(i)}|Y_\ell^{(j)})] \approx \frac{1}{k} \sum_k P(Y_k^{(j)} = 1|Y_k^{(i)}) - P(Y_k^{(j)} = -1|Y_k^{(i)})$$

which we can estimate using $f(1 - J(Z^{(i)}, Z^{(j)}))$ where J is the Jacard distance; i.e., we estimate

$$E_\ell[P(Y_\ell^{(i)}|Y_\ell^{(j)})] \approx a_{ij} := f\left(\frac{\sum_k 1_{\{Y_k^{(i)}=Y_k^{(j)}\}}}{\sum_k 1_{\{Y_k^{(i)}\neq 0 \text{ and } Y_k^{(j)}\neq 0\}}}\right).$$

where $f : [0, 1] \to [-1, 1]$ monotonically. However, sample expected value is highly subject to outliers. To mitigate this issue and to estimate the values for f, we hierarchically cluster the users using the Jacard distance and let $f = 1$ for i and j in the same cluster and let $f = 0$ elsewhere. Note that $a_{ii} = 1$.

Therefore, we estimate

$$P(Y_n^{(i)}) = \frac{1}{\sum_j a_{ij}} \sum_j a_{ij} P(y_n^{(j)} = 1|X_n^{(j)}).$$

In practice, we use log-liklihood $\ell(\theta)$ and Laplace smoothing to classify by selecting the larger of:

$$\log P(Y_k^{(i)} = 1|X_k^{(i)}) = \log P(Y_k^{(i)} = 1) + \sum_{w_j \in X_k^{(i)}} \log(\phi_{w_j|Y^{(i)}=1}) \text{ and}$$

$$\log P(Y_k^{(i)} = -1|X_k^{(i)}) = \log P(Y_k^{(i)} = -1) + \sum_{w_j \in X_k^{(i)}} \log(\phi_{w_j|Y^{(i)}=-1})$$

where $\phi_{w_k|Y^{(i)}=1} = \frac{\sum_j a_{ij}\eta_k^{(j)}+1}{\sum_j a_{ij} \sum_h \eta_h^{(j)}+p}$ and $\phi_{w_k|Y^{(i)}=-1} = \frac{\sum_j a_{ij}\hat{\eta}_k^{(j)}+1}{\sum_j a_{ij} \sum_h \hat{\eta}_h^{(j)}+p}$.

4 Experimental Proceedure

Testing was performed using k-folds cross validation running our algorithm on the dataset for $k = 2, 4, 7, 10$ to predict the tweets for an invidual user (holding fixed and training on the values of the other users for clustering purposes). We ran our algorithm using cluster sizes of 1, 6, 9, 16, and 25. For a control set, we also ran the same experimental proceedure using the stock Bayesian classifier in the NLTK toolkit.

5 Results

In comparison with the NLTK Naïve Bayes, our algorithm outperformed the toolkit in Positive Precision (73.98% to 62.67%), in Positive Recall (64.76% to 36.59%), and Negative Precision (85.50% to 76.06%), while the toolkit implementation very slightly outperformed our algorithm in Negative Recall (90.23% to 90.12%). In overall accuracy, our algorithm substantially outperformed the toolkit Naïve Bayes, 82.46% to 73.64%.

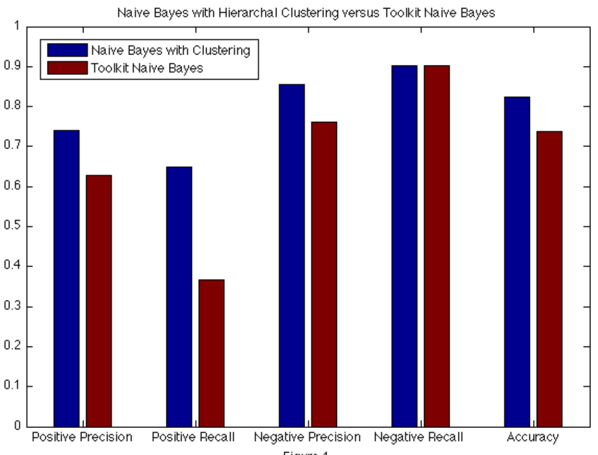


Figure 1

Our performace by fold was largely consistant with a slight bump on 2-fold cross validation (by approximately 84.3% to 78.9%. Results were also most positive with 6 clusters, achieving an accuracy of 84.25%, compared with 76.55% for 1 cluster, 80.14% for 9 clusters, 76.28% for 16 clusters, and 68.45% for 25 clusters.

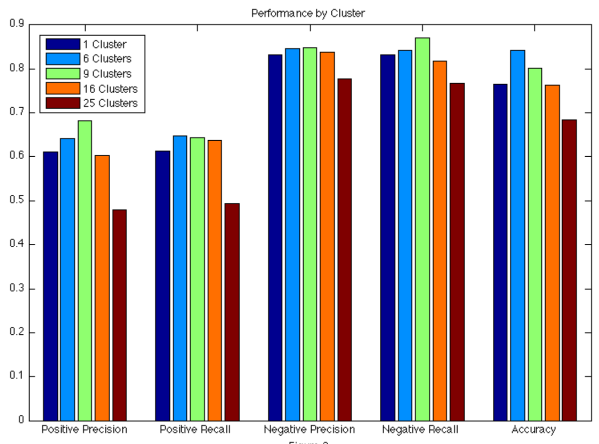


Figure 2

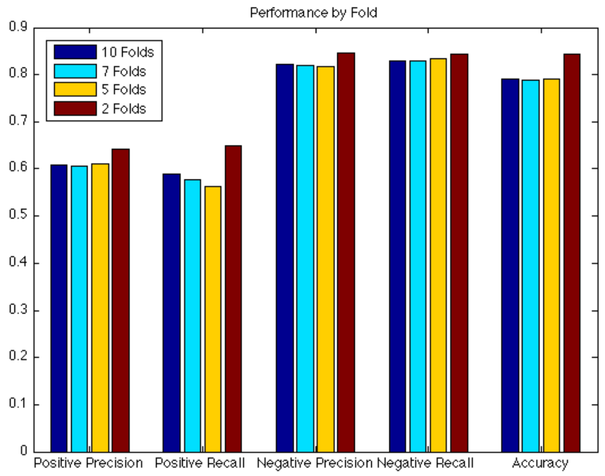


Figure 3

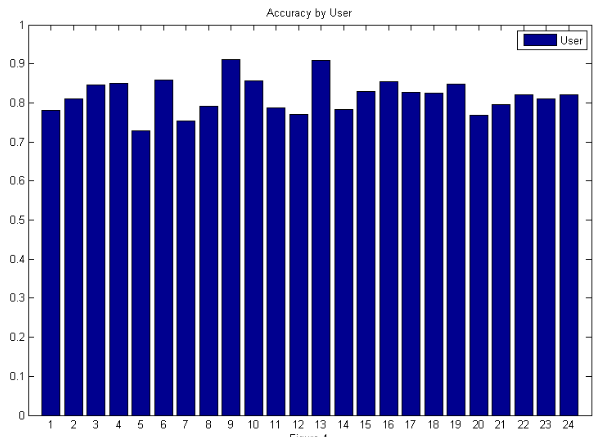


Figure 4

6 Conclusion

The overall accuracy of the collated classifier was around 75-85% based on the average results of all 25 users. The simple Naïve Bayes classifier from the NLTK natural language processing package achieved around 70%. The advantage of our classifier over the toolkit implementation lies on the collated nature of the classifier which strengthens the classifications by bringing in extra information for each users base Bayesian classifier. In particular, our algorithm significantly outperformed the NLTK toolkit classifier on positive recall, which is arguably the most significant metric with regards to user experience.

Examining overall trends, our classifier performed very well on precision and recall metrics for disliked tweets, but significantly worse on the same metrics for liked tweets. There are a number of reasons for this disparity between classifying likes and dislikes. Ratings were more heavily weighted towards dislikes as an overall trend by raters, with simply the pure number of dislikes heavily outweighing the number of likes. Therefore, there were much fewer examples of liked tweets to train the Bayesian classifier to begin with. Furthermore, users were much more likely to dislike entire blocks of tweets simply based on author whereas the qualities exhibited for a liked tweet are much more nuanced, with only a small fraction of tweets even from the users favorite tweeters being worthy of being liked. This makes it difficult to predict whether a user will like a tweet, resulting in poor performance in predicting liked tweets. Nevertheless, the overall accuracy of our classifier is fairly high. Our test error was actually smaller for 2-folds cross validation, indicating that the modeling each word (with relevant preprocessing) as a feature likely overfits the data. Future research will focus on improving the choice and quality of features, which should additionally improve the overall accuracy. The consistancy across folds did indicate success in earlier prediction of tweets, which as introduced with the problem, is an important factor in a successful algorithm. Given the difficulty in classifying tweets because of their short nature and lack of very much informational content, the classifier did a good job at determining whether a user would like or dislike a tweet.

References

[1] Mock, Kenrick. Dynamic Email Organization via Relevance Categories. Intel Architecture Labs. May 6, 1999.

[2] Mock, K., Vemuri, V. Information Filtering via Hybrid Techniques. *Journal of Information Processing and Management*, Permagon Press, v33, n5, pp 633-644. 1997.

[3] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. A Bayesian approach to filtering junk e-mail. AAAI'98 Workshop on Learning for Text Categorization, Madison, WI, July 1998.