Author Identification on Twitter

Antonio Castro
antonio.alfredo.castro@gmail.com

Brian Lindauer
brian@shendauer.com

I. Introduction

As of June 2012, Twitter had 500M users, 140M of whom are in the United States. These users, especially those outside of the United States, may assume that they have a certain level of anonymity among this sea of tweets. Our project investigates whether the identity of an anonymous Twitter user can, in fact, be uncovered using only linguistic stylometry. Authorship recognition is a very well-studied domain, but the scale is almost always limited to no more than a few hundred authors. Narayanan, et al. [2] study authorship recognition at Internet scale, by looking at the characteristics of different classifiers when applied to a corpus of approximately 100k blogs. We set out to answer the question of whether similar results could be achieved on tweets rather than blogs, despite their much shorter length. Starting with the set of features, classifiers, and normalization methods that yielded the best results over blog data in [2], we adapt them to Twitter data and measure the results.

II. Data Collection

Our set of training data spans more than 800 users, most having at least 1000 tweets per user. This provides enough data to our classifiers despite the sparsity of the derived features and the limitations imposed on the length of a tweet. Our data set also includes a number of Twitter accounts where we have prior knowledge that they are authored by the same user as at least one other account in the set.

Table I describes the various sources we reviewed and how we ended up using the data. We initially utilized a web site called discovertext.com to follow the desired accounts and begin collecting data. However, the service is insufficient for our needs because it fails to capture information about retweets and is unable to provide historical tweet data. As a result, we utilize the Twitter API directly to gather the last 1000 tweets of each of the users identified in Table I. Primarily because of Twitter rate limit limitations, the data gathering requires several days to complete.

Our experimental methodology requires that we identify a collection of Twitter account pairs where one author is responsible for both accounts. Our primary methods for identifying those accounts where issuing a request to employees of Dell with official Dell Twitter accounts, and using Google to search for phrases indicating multiple Twitter accounts.

Source	Desired Usage	Actual Usage
Klout	Top thousand klout users as a source of a variety of users with rich content.	Not viable for use. Only small lists of top Klout rankings are published.
Twitter Firehose	Obtain tons of tweets across a massive number of users.	Not viable for use. While the data is broad, it does not provide us with a long enough history of any individual user to populate our sparse set of features.
Twitaholic	Provides a top 1000 most followed list of Twitter users.	Provides the basis of our twitter users to gather data from.
Dell Solicitation	Obtain a list of employees who maintain two separate Twitter accounts.	Provides a set of known Twitter accounts that are authored by the same person.
Web Search	Search for phrases such as also follow me at on Twitter profile pages.	Provides a set of known Twitter accounts that are authored by the same person.
Google Plus Profile Scrape	Discover users that report having multiple Twitter feeds to be followed.	Adds to set of Twitter accounts with known authors.

Table I
Data Sources

We also obtained some meta-information about Google Plus profiles from the authors of "How Unique and Traceable are Usernames?." [4] and used that to target a crawl of Google Plus profile data. This produced an additional set of account pairs. After eliminating non-English feeds, feeds containing only links, etc., we are left with 58 labeled accounts from 27 different authors.

Due to the small number of valid account pairs in which we have prior knowledge of the account being authored by the same user, we simulate dual authorship by splitting each feed 70/30 and performing cross-validation and measuring the error. Since this measures error against the same feed, rather than another feed by the same author, it is not ideal,

but in most cases, it should approximate a lower bound, and as such it gives us a good indication of the algorithm's general performance in de-anonymizing tweets.

Additionally, some accounts exhibit issues that might introduce confounds into the experiment. Examples of these include foreign languages and tweets automatically generated by applications, such as Foursquare. We do not incorporate any cleansing of these issues in our experiments other than the final test we performed using the known accounts with duplicate authors. For those accounts, we manually review their feeds to remove any feeds that contained any obvious issues.

The results in this paper are based on a collection of 844 Twitter streams containing approximately 777k total tweets. These include the 27 sets of accounts that are known to be maintained by the same author.

III. Features

Our selection of features is inspired by Narayanan [2], Writeprints [1], and Ireland [3]. It is, in fact, mostly a subset of the Narayanan features. These features aim to focus on the style of the tweet rather than its topic. So, for example, we specifically look at function/stop words rather than ignoring them and focusing on words with large TF.IDF scores. In adapting Narayanan's previous work to Twitter, we add several Twitter-specific features, described in Table II.

Category	Description	Count
Length	words/characters per post	2
Word shape	frequency of words in uppercase, lowercase, capitalized, camelcase, and other capitalization schemes	5
Word length	histogram of word lengths from 1-20	20
Character frequencies	frequency of letters a-z (ignoring case), digits, and many ASCII symbols	68
Unicode	frequency of non-ASCII characters	1
Function/stop words	frequency of words like "the", "of", and "then"	293
Twitter conventions	existence of "RT" or "MT"	2
Retweets	whether the post is an exact retweet, or a modified retweet	2

Table II
Features used, adapted from [2]

We extract all 393 of these features using a Ruby script, and store them in CSV for ingestion by the classifier programs, which are implemented in MATLAB. Extracting these features presents no notable challenges in scalability or algorithmic complexity.

We were curious to know which of these features has the most impact on our classification accuracy. To find out, we adoped the definition of information gain used in Narayanan. That is,

$$IG(F_i) = H(T) - H(T|F_i) = H(T) + H(F_i) - H(T, F_i)$$

where H is the Shannon Entropy, T is the random variable for the Twitter account number, and F_i is the random variable for feature i [2]. By using the same information gain metric as Narayanan, we are also able to compare the effective features in detecting blog authorship to those in detecting Twitter authorship. The most influential features for our classifier are listed in Table III.

Feature	Information gain (bits)
Freq. of non-ASCII characters	0.60209
Number of words per tweet	0.46262
Freq. of all lowercase words	0.3711
Freq. of o	0.35851
Number of characters per tweet	0.3576
Freq. of a	0.32181
Freq. of t	0.31425
Freq. of e	0.29231
Freq. of .	0.29063
Freq. of words with only the first letter capitalized	0.28206
Freq. of h	0.25791
Freq. of n	0.25051
Freq. of i	0.24
Freq. of r	0.2162
Freq. of @	0.21448

Table III
Features with highest information gain

Comparing this list with Narayanan's top 10 features for blogs, we see that both find the length of the post and the capitalization style to be highly discriminative. However, the most discriminative individual characters are different for blog posts and tweets. Narayanan found that apostrophes, periods, and commas were the most important characters for blogs, while we found that o, a, t, e, and period were the most important for tweets. This result was initially puzzling, but we soon realized that all of the characters comprising http:// are near the top of our list. It may be that vowels are highly ranked becuase they help distinguish tweets consisting primarily of English words from those containing mostly URLs.

Far and away, our most influential feature is the frequency of non-ASCII characters. Some Twitter users routinely include unicode in their tweets, while others never do. Since we are nominally filtering out non-English accounts, these unicode characters are mostly special characters, such as

hearts and smilies, rather than characters in non-English words.

Finally, we note that the frequency of "@" is highly ranked, but not in our top ten. This character is used on Twitter to denote a reference to another Twitter feed. It far outperforms our more Twitter-specific features, such as the presence of "RT" (0.1 bits), or whether a retweet includes an exact/inexact copy of the original tweet (0.06 and 0.04 bits, respectively).

IV. Classifiers

Because Narayanan, et al. reported their best results with a combination of nearest neighbors (NN) and regularized least squares classification (RLSC), we implement these classifiers and run them against the data.

Initially focusing on NN, we implement the variation described by Narayanan along with the normalization procedure from that same paper. Rather than keeping all data points in memory, which would be very expensive considering the number of Twitter users, we compute one centroid for each Twitter account. To do this, we read in all extracted tweet features from all users, then normalize by both column and row. First, each column value is normalized by the mean of the non-zero values in that column. Then, each row value is divided by the norm of that row.

At prediction time, we read the extracted features of each tweet in the test stream. For each of those tweets, we measure the Euclidean distance to each of the centroids computed in training. Then we take the sum of the distances of all the tweets to each of the centroids and rank the centroids by their average nearness to a tweet in the test stream. We ask whether the account by the same author appears in the top N% of the ranking, including whether it is ranked first. We also measure generalization error using 70/30 cross validation on tweets from the same account. In computing both training error and cross validation generalization error, we count a prediction as correct only if the correct account appeared first in the ranked list. With our training and cross validations set, NN yields a 0.61% training error and a 2.5% generalization error. This generalization accuracy is surprising considering the relative simplicity and lack of domain specificity in our featureset. In fact, the feature set is composed largely of one-grams and function words.

Though our sample size is small, we are able to use our 58 labeled accounts to get a measure of accuracy when the algorithm is applied to our intended use case. Given one of the accounts in the pair, the NN classifier ranks the other account first 29% of the time – an error rate of 71%. But the correct author is ranked at least 2nd 38% of the time. And they appear in the top 10% of the ranking over 70% of the time. Figure 1 shows the cumulative distribution over these rankings.

This NN algorithm is relatively fast, since it only needs to compare each test point to one centroid for each training class. Because these tests are independent and lightweight, we are able to utilize MATLAB's parallel computing capabilities and process the entire dataset in minutes.

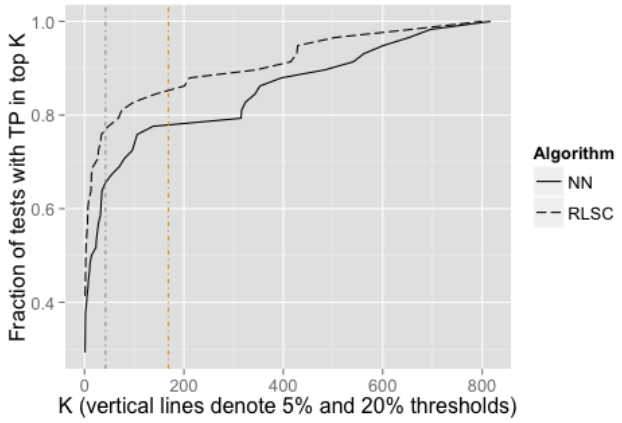

Figure 1. Cumulative distribution of percentile rank over test examples

We implement the Regularized Least Square Classifier algorithm starting with the original closed form solution as described in Rifkin's paper [5].

$$W = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

Given the size of the training set of approximately 500,000 tweets in our design matrix X, and the multiclass {-1, +1} convention Y matrix with 844 classifier vectors the algorithm requires approximately 12 hours to complete for each classifier vector. However, we found that the training error performance is not affected by converting to a {0,1} convention and we are able to complete the entire classifier calculation using sparse matrices in MATLAB in a single step

$$W = (X^T X + \lambda I)^{-1} X^T Y$$

within a few minutes.

As a matter of convention, we classify each tweet as either a positive or negative by determining the maximum value of each linear classifier method for a single tweet, setting that classifer to one and the others to zero. We used the frequency of a classifier being chosen across the sample set as an aggregate to determine confidence accross a sampling of tweets. However, this result does not vary notably from using the sum of the raw values produced by each tweet.

Similar to the results in the Narayanan paper, the resulting training error of this One Vs. All (OVA) method, ranges between 20% and 30% which is more than an order of magnitude worse than our NN classifier. This is a known problem for OVA classifiers with heavily skewed data. For each label in our data set, the RLSC generally has a thousand positive examples compared to nearly half a million negative examples. We explored a handful of methods to level the

data as the algorithm. Rescaling the alone is not an option as the algorithm is invariant to rescaling of the input data. We try a few methods [7] to reduce this error by including pruning [6] negative examples for each OVA being trained and artifically creating like samples for training. However, we find that these methods only improved the classification error by only $5 - 7\%$.

Inspired by the penalization concept in the Narayanan paper, we set out to modify the cost function so that it penalizes false negatives proportionally more heavliy than false positives. The closed form solution for appropriatley balancing the ratio of positive and negative examples is as follows. We let Φ_j be an $m \times m$ diagonal matrix where $\Phi_{j_{ii}}$ is equal to the ratio of negative to positive examples in rows corresponding to a positive example in classifier j, and 1's for all other diagonal values.

More formally:

$$w_j = \frac{\sum_i 1\{y_{j_i} = 0\}}{\sum_i 1\{y_{j_i} = 1\}}$$
$$\Phi_{j_{ii}} = w_j y_i + (1 - y_i)$$

Using these classifier specific Φ_j and $\vec{y_j}$ values, we can derive θ_j for each classifier j in closed form as follows.

$$J(\theta_j) = \frac{1}{2}(X\theta_j - \vec{y_j})^T \Phi_j (X\theta_j - \vec{y_j}) + \lambda ||\theta_j||^2$$
$$\nabla_{\theta_j} J(\theta_j) = X^T \Phi_j X\theta_j - X^T \Phi_j \vec{y_j} + 2\lambda\theta_j = 0$$
$$\theta_j = (X^T \Phi_j X + 2\lambda I)^{-1} X^T \Phi_j \vec{y_j}$$

Using this closed form requires the generation of $\Phi \in \Re^{500kx500k}$ for each of the 844 classifiers and requires iterating through each classifier independently. Due to the memory requirements of this algorithm in MATLAB, even using sparse matrices, this closed form algorithm is difficult to parallelize beyond a handful of threads and thus still required several hours to complete. However, with these results we are able to reduce our training error to 1.1% and our generalization error to 4.7%. Most importantly, the algorithm performs significantly better than NN on the set of known Twitter dual accounts, classifying 41% of them correctly. Comparing RLSC to NN in Figure 1, we see that RLSC outperforms NN at every threshold.

In accomplishing the above results, we do not experiment much with the regularization paramater λ. Our choice of lambda is the minimum order of magnitude that did not result in the classifiers calculation resulting in nearly singular matrices for the algorithm, which, in this case was 10^{-9}. We minimally explore methods to improve the performance and memory requirements of the algorithm without notable impact, but do not explore methods that do not require large matrix operations such as conjugate descent and dynamic programming as referenced by Naranayan and Rifkin [5] which may be necessary to scale significantly beyond the number of users we are working with in this experiment.

V. Recommendations

Given the prominant use of Twitter by political dissidents, it's alarming to learn that the author of an "anonymous" feed might be identified with 40% accuracy based only publicly viewable information. By using other information, such as the time of posts and IP access logs, the identification rate is likely to be much higher. However, having examined many of the failed matches, we note that evasion is simple, as long as the author is aware of these risks. In several of the failed matches, one of the accounts is written in a deliberately different voice. In one case, the author tweets in the voice of his dog. In other cases, the author of limits the amount of commentary in the second feed and mostly posts URLs. By following these examples and either deliberately altering their writing voice, or limiting the amount of text posted, Twitter users can help maintain their anonymity.

VI. Future Work

Given the initial promising results, there is opportunity for continued work in testing classifier accuracy and performance at larger scales, as well as in several other areas, including:

- Expanding the exploration of stylometry features in previous work to further impact generalzation error.
- Collecting a broader set of known account pairs and/or identifying a list of potential pairs to be confirmed.
- Exploring conjugate descent, dynamic programming, or stochastic methods to improve the speed and memory usage of the RLSC algorithm.
- Exploring ensemble learning to combine RLSC and NN algorithms and its impact on generalization error.
- Applying these methods on a much larger number of accounts (e.g. 1,000,000 twitter accounts).
- Removing language and application data that may be introducing confounds.

VII. Conclusion

We are encouraged by our results. The classifiers both exhibit excellent accuracy in cross validation, and do fairly well in the general use case where the test stream comes from a different Twitter account by the same author. We expected less accuracy from Twitter than from the blog or email data used in previous work because the size of the tweets are significantly smaller in comparison. With the small number of labeled examples in our data set, we cannot make a judgement about whether performance was better or worse on Twitter vs. blog data, but we can conclude that stylometric analysis does, in fact, perform well on tweets.

References

[1] Abbasi, Ahmed, and Hsinchun Chen. "Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace." ACM Transactions on Information Systems 26.2 (2008): 7.

[2] Narayanan, Arvind, et al. "On the feasibility of internet-scale author identification." Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012.

[3] Ireland, M.E., & Pennebaker, J.W. (2010). Language style matching in writing: Synchrony in essays, correspondence, and poetry. Journal of Personality and Social Psychology, 99.

[4] Perito, Daniele, et al. "How unique and traceable are usernames?." Privacy Enhancing Technologies. Springer Berlin/Heidelberg, 2011.

[5] Rifkin, Ryan, Gene Yeo, and Tomaso Poggio. "Regularized least-squares classification." Nato Science Series Sub Series III Computer and Systems Sciences 190 (2003): 131-154.

[6] Ofer Dekel Ohad Shamir. "Multiclass-Multilabel Classification with More Classes than Examples." Proceedings of the 13th international Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

[7] Schapire, Robert E., and Yoram Singer. "BoosTexter: A boosting-based system for text categorization." Machine learning 39.2 (2000): 135-168.