Author Attribution of Yelp Reviews

Emma Strubell Patrick Verga

Abstract

We attempt to identify the authors of Yelp reviews using two versions of k-means clustering and support vector machines.

1 Introduction

Author attribution, the ability to identify the author of a given document given its text, has implications for companies trying to identify fraud, law enforcement identifying posts by criminals or terrorists, and even historians working to discern the true creator of a disputed piece of text. These principles are similarly important for individuals seeking privacy or anonymity, such as dissidents of a repressive government.

Companies such as Yelp, TripAdvisor, and Amazon who provide product or business reviews as a service to their customers rely on having accurate and truthful reviews of those products and businesses. However, companies or individuals may artificially inflate their own review score or deflate those of their competition. The ability to accurately and effectively identify and remove fraudulent reviews would be extremely beneficial to these companies to ensure the integrity of their product. In our work we attempt to apply principles of stylometry and author identification to the problem of fraud detection.

2 Background

Stylometry is the analysis of individuals usage of linguistic features to attempt to identify those individuals. In some cases this can include standard information retrieval techniques such as term frequency-inverse document frequency (TF-IDF) bag-of-words approaches. However, using bag-of-words has the downside of capturing context specific variability rather than writing style variability. That is, differences in what the piece is written about rather than the way the piece is written.

There are several different ways to approach the problem of author identification. They can be broadly separated into two groups: classification and similarity detection. The more widely studied problem is that of classification. Given a set of authors each with a set of documents, one can train a model to then attribute a new unknown document to one of the known authors. The less-investigated problem is that of similarity detection.

Here, instead of using a training corpus to define a small number of authors, one can use a technique such as clustering to partition a set of documents into different clusters that should correspond to different authors.

There is a great deal of existing research in the field of stylometry, some even predating computers; one of the earliest applications was attempting to discern the true author of the Federalists papers. More recently, researchers have attempted to classify authors of twitter posts [1, 2] and web forum posts [3, 4]. [5] attempted to use topic modeling for author attribution.

Most attempts to discern authorship are performed on a very small set of authors each given a large corpora of training data. Some work has been done on internet scale problems given thousands or an unknown number of authors. [6] attempted to classify posts of several thousand authors with moderate success. Their work was based off of the Writeprints model [7] whose feature choices have become very influential. We crafted our feature set based on [6].

3 Dataset

Yelp's Academic Dataset[1] provides data and reviews for the 250 closest businesses to 30 universities in North America. We used the subset of the data for the city of Pheonix, Arizona. The data consists of three different types of JSON object: user objects, business objects, and review objects. For our task we were interested in the `user_id` and `text` properties of the review objects, as well as the `review_count` property of the user obejcts for further pruning of the data. In our actual experiments, we used the reviews of the 23 users who had written more than 250 reviews in an attempt to provide our learning algorithms with enough data to correctly classify the documents. Although the Yelp dataset provides additional information about the users and reviews that might be useful in user review classification, such as user star ratings and the date the review was posted, we elected not to use this data as features in order to avoid tuning our approach too exclusively to the review domain. Instead, we limited our features to those based on the review text only, and thus that could be applied to achieve author attribution on text from any domain, such as forum posts, blogs or emails.

4 Methods

4.1 Features

As noted in the previous section, we limited our features to those based on the text of the review. Based on [6], the features we used are listed in Table 1.

Syntactic features such as part-of-speech tags and dependency labels were generated using the natural language processing pipline included in FACTORIE [8]. In our compu-

[1] `https://www.yelp.com/academic_dataset`

Feature Type	**Description**
Document length	Number of words, characters in post
Vocabulary richness	Yule's k
Word shape	Frequency of capitalization types (all, none, first, camel)
Word length	Frequency of words containing 1–20 letters
Letters	Frequency of letters a–z (case insensitive)
Digits	Frequency of digits 1–9
Punctuation	Frequency of `.,!?;:()"'`
Special characters	Frequency of `~‘@%#$%^&*_+=\{\}[]\\|/<>`
Function words	Frequency of function words; See text
Part-of-speech	Frequency of 50 different part-of-speech tags
Syntactic dependency label	Frequency of 50 different syntactic dependency labels

Figure 1: Table of features used in our model.

tation of Yule's k measure for vocabulary richness, we used standard WordNet [9] lemmas. We used the following equation to compute Yule's k:

$$k = 10000\frac{m_2 - m_1}{{m_1}^2} \tag{1}$$

where m_1 is the number of distinct lemmas found in the document, and m_2 is defined in terms of f_i, the frequency of lemma i in the document, and c_i, the count of lemma i in the document:

$$m_2 = \sum_{i=1} c_i f_i^2 \tag{2}$$

In initial clustering experiments using bag-of-words features, we found that including frequencies of all words found in the text as features caused the algorithms to group reviews based on topic rather than author. This makes sense since the occurence of the word "spicy" in restaurant reviews is more likely to be a commonality of Mexican restaurants than of an individual author across restaurant types. So, we chose not to include features for all words in the text. Instead, following the technique in [6], we used frequencies of all function words observed in the text. Function words are words such as "hers," "of," "to" and "therefore," the types of words usually used as stop words (and thus removed from consideration) in document topic modeling. The idea is that patterns of use of these words would be a better indicator of document authorship since they better represent an individual's habits of writing rather than the topic of the individual's words, as would their use of certain nouns, for example. Since the set of words that make up function words is not well defined, we used words that were tagged with parts-of-speech from the list given in Table 2, which were based on the Wikipedia page for function words[2] combined with the set of 50 part-of-speech tags used by our tagger, and common sense.

[2] `http://en.wikipedia.org/wiki/Function_word`

POS Tag	Description
CC	Coordinating conjunction
DT	Determiner
EX	Existential there
IN	Preposition or subordinating conjunction
LS	List item marker
MD	Modal
PDT	Predeterminer
POS	Posessive ending
PRP	Personal pronoun
PRP$	Posessive pronoun
RP	Particle
TO	To
UH	Interjection
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Posessive wh-pronoun

Figure 2: Parts-of-speech used to determine function words. The frequency of each word tagged with one of these parts-of-speech was used as a feature.

4.2 Clustering

We performed clustering using the Apaches Mahout library[3] for scalable machine learning and data mining. We performed k-means clustering with both the standard randomized initial centroids as well as centroids estimated by a canopy algorithm.

4.2.1 k-means

k-means is the most widely used clustering algorithm due to its balance of simplicity and accuracy. In this method, the user specifies k, the number of clusters the algorithm will assume to exist in the data. The algorithm is initialized by choosing k random centroids. At each iteration, every input vector is assigned to the closest centroid. After each point has been assigned, each centroid is re-evaluated to be the mean of each point assigned to it. This continues until some error convergence metric is satisfied.

4.2.2 Canopies

A canopy algorithm is a fast clustering algorithm that quickly estimates k as well as the initial cluster centroids. Here the user specifies two values, $t_1 > t_2$, defining two distances. For each point, if it is less than t_2 from any defined centroid it will join the cluster and be

[3] `http://mahout.apache.org/`

removed. If the point is less than t_1 away from a centroid it is assigned to the centroid but is free to join other clusters. If it is greater than t_1, it forms the centroid of a new cluster. While the final result of the canopy algorithm is not as effective as a slower algorithm like k-means, it is very effective as a bootstrapping step to estimate the initial values for k-means.

4.3 Support Vector Machines

We performed multi-class classification of our data using support vector machines (SVMs) using the SVM-Light library for multi-class classification[4]. An SVM is a binary classifier that is trained to linearly separate unknown examples into one of two categories. Generally, multi-class classification is achieved by training an SVM to distinguish between each pair of classes, called one-vs-all training. Each new example is then sent through each SVM and the most likely class is chosen.

We used SVMs with a linear kernel because we found that more complex polynomial kernel functions took far too long to compute, and SVM-Lights implementation of multi-class classification with a radial basis function (RBF) apparently did not work (the program refused to learn support vectors. Since the documentation states that non-linear kernels are relatively new functionality in the multi-class classification package, we suspect that this functionality is simply not yet implemented, and the documentation is mislead us by saying that it is.)

Linear SVMs, on the other hand, were extremely fast compared to their nonlinear counterparts since they can be implemented using quadratic programming to solve an optimization problem, as described in [10], eliminating overhead of considering every possible pair of classes.

5 Results

5.1 Clustering

We ran k-means clustering on 7750 posts by 23 authors using k values of 23, 100, 1000, 1800, and 4000. We ran the clustering algorithms with both randomized initial centroids and with centroids derived from the canopy algorithm. Our graphs show the average recall, precision and F score for each cluster.

The next graphs show the difference in precision, recall, and F scores between k-means run with the randomized centroids and canopy centroids.

5.2 Support Vector Machines

Multi-class classification using non-linear SVMs is extremely slow on high dimensional data. We therefore were only able to obtain results for linear SVMs. For the experiment

[4] `http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html`

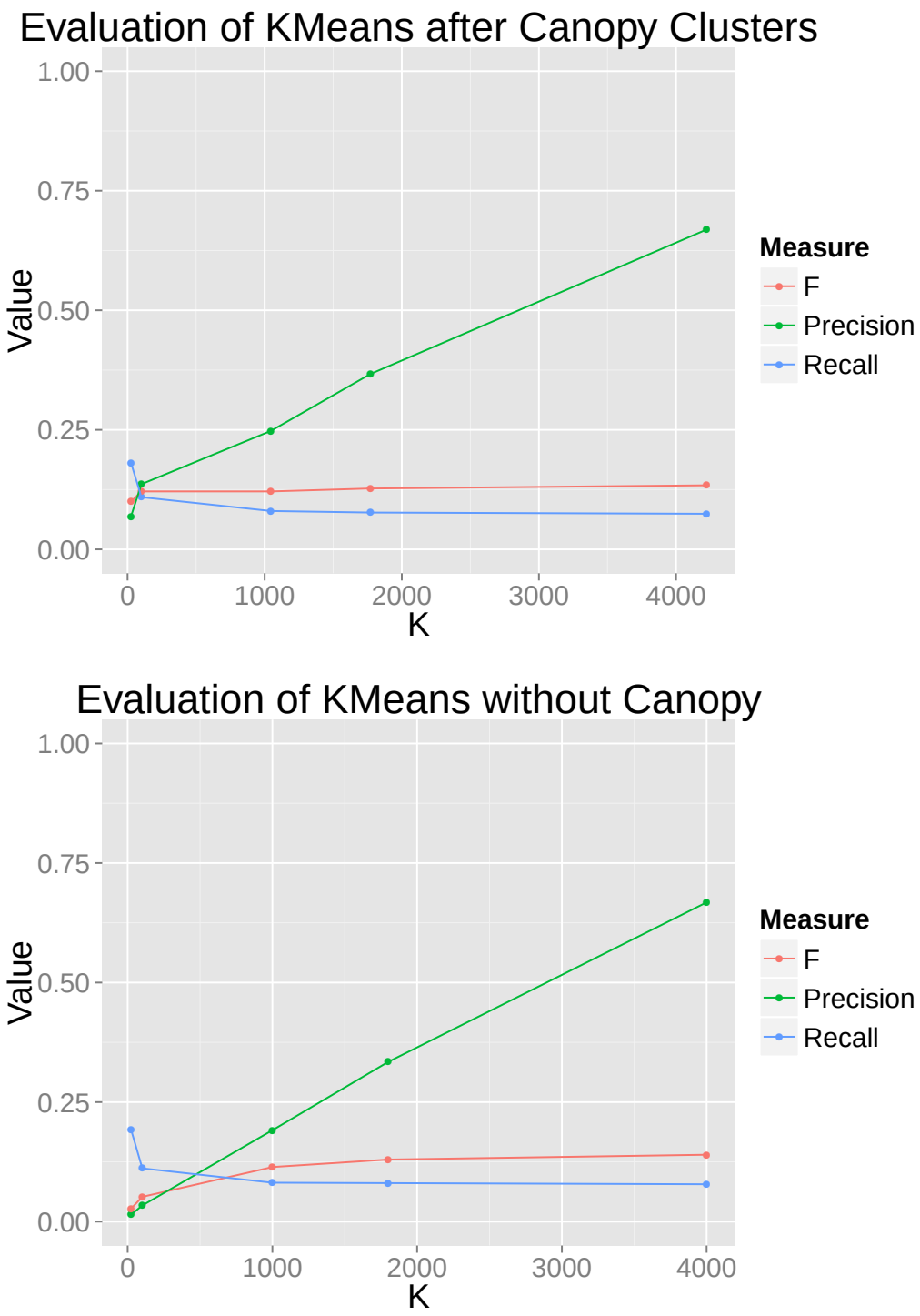

Figure 3: The effect of varying k in k-means clustering on F score, precision and recall with and without canopies.

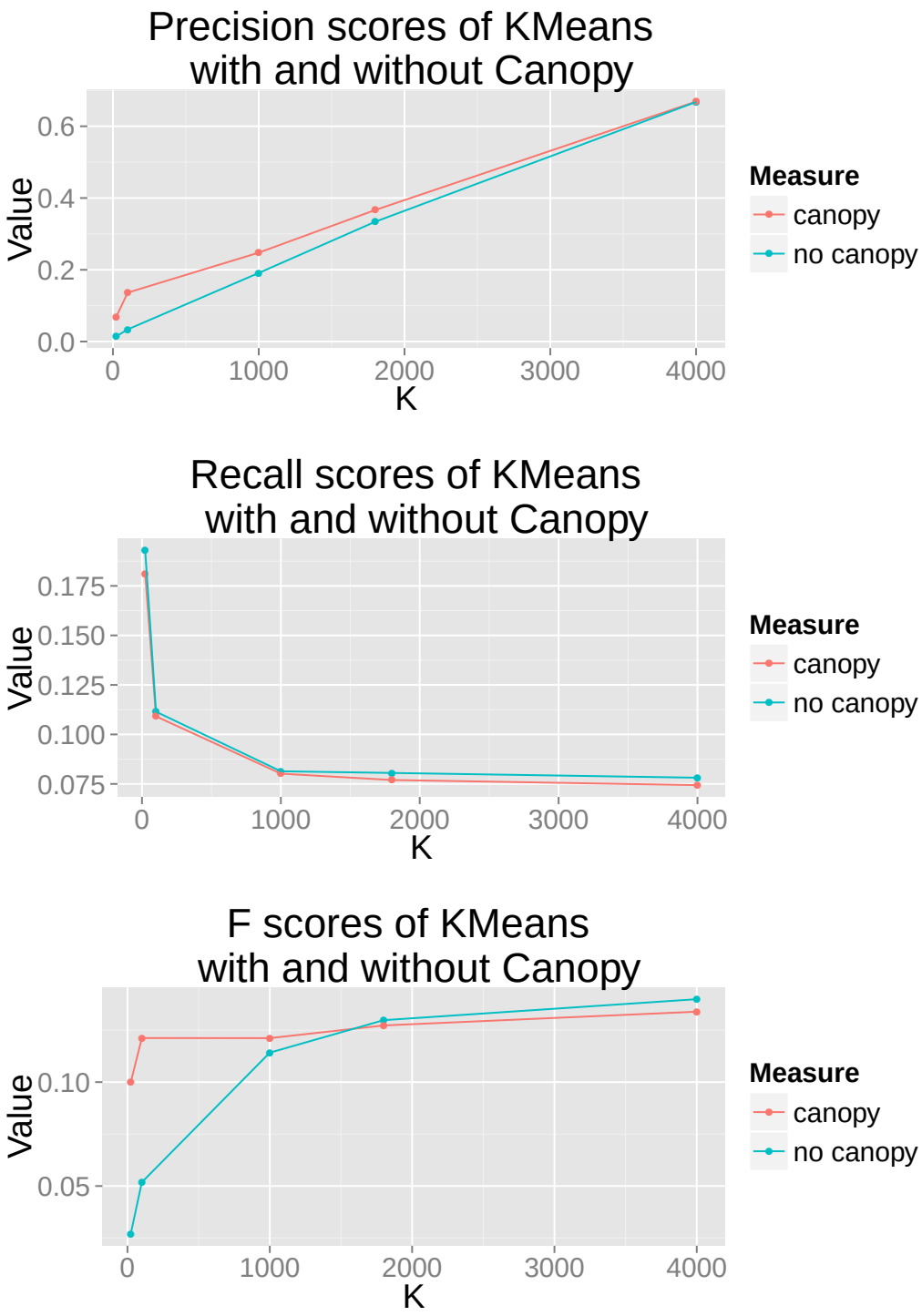

Figure 4: The effect of bootstrapping using canopies to precision, recall and F score.

C	**Error**
1.0	91.47
2.0	90.83
3.0	90.24
4.0	89.65
5.0	89.94
6.0	89.78
7.0	90.32
8.0	89.94
9.0	**89.49**
10.0	89.50

Figure 5: The effect of varying C, the tradeoff between margin size and training error, on classification error by a linear SVM on 23 classes.

with 23 classes, results listed in Figure 5, we split the 7750 examples into 6500 training and 1250 testing examples. The results are averages over 10 trials of splitting the data into 7750/1250 test/train split at random (without replacement).

Clearly, the error rates are high, though above random considering that we are classifying into 23 different classes. Varying C, the tradeoff between margin size and training error, made little difference. We also tried varying C in increments of 10 from 10 to 100, which also made little difference.

Seeing these high error rates, we tried running simpler experiments with only 5 classes. In this case we split the data into 1556 training and 300 testing examples, similar relative proportions to the experiment with 23 classes. We did the same averaging over 10 random splits in this case.

Here we see greatly improved results, hovering around 50% error, which again is better than random since we are classifying into 5 different classes. There is a stronger trend in this data of results increasing as C increases, but further experimenting with C did not provide significant improvements.

6 Discussion

The problem of author identification is a very difficult machine learning problem. Even humans can have a difficult time discerning the authorship of an unknown piece of text. Our initial results leave much room for improvement and future directions of research. The biggest boost in performance would most likely come from further feature engineering, or extracting higher order features from the data such as bigrams, or syntactic dependency pairs. It seems like our features failed to encapsulate the properties of an individual's writing that distinguishes him or her from others. Although we implemented a widely used set of features, there is no definitive set of features for author attribution classification. It

C	**Error**
1.0	55.50
2.0	56.23
3.0	54.57
4.0	52.73
5.0	53.77
6.0	53.93
9.0	54.53
8.0	52.16
7.0	52.73
10.0	**52.13**

Figure 6: The effect of varying C, the tradeoff between margin size and training error, on classification error by a linear SVM on 5 classes.

is also possible that other classification algorithms may fare better on this task than the support vector machines or clustering algorithms that we tried.

References

[1] Antonio Castro and Brian Lindauer. Author Identification on Twitter. 2012.

[2] Robert Layton, Paul Watters, and Richard Dazeley. Authorship Attribution for Twitter in 140 Characters or Less. *2010 Second Cybercrime and Trustworthy Computing Workshop*, pages 1–8, 2010.

[3] Sangita R. Pillay and Thamar Solorio. Authorship attribution of web forum posts. *2010 eCrime Researchers Summit*, pages 1–7, 2010.

[4] Richmond Hong Rui Tan and Flora S. Tsai. Authorship Identification for Online Text. *2010 International Conference on Cyberworlds*, pages 155–162, 2010.

[5] Jacques Savoy. Authorship attribution based on a probabilistic topic model. *Information Processing & Management*, 49(1):341–354, 2013.

[6] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the Feasibility of Internet-Scale Author Identification. *2012 IEEE Symposium on Security and Privacy*, pages 300–314, 2012.

[7] A Abbasi and H Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems (TOIS)*, 2008.

[8] Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.

[9] Christiane Fellbaum. WordNet and wordnets. In Keith et al. Brown, editor, *Encyclopedia of Language and Linguistics, Second Edition*, pages 665–670. Oxford, 2005.

[10] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs. *Journal of Machine Learning Research*, 2001.