Identifying Internet Users by Writing Style

Adam Novak

December 11, 2012

Abstract

An experiment was carried out to evaluate the author-identification results of Narayanan *et al.* when extended the problem of re-identifying single author-labeled comments from the Internet discussion site Reddit.com [8, 13]. A data set of approximately 70,000 comments from 100 users was analyzed. Both bag-of-words and stylometric (or "content-free") methods of feature extraction were explored, in combination with the 2-Nearest-Neighbor, Naive Bayes, and SVM classification algorithms. Overall, the most accurate combination proved to be Naive Bayes and a bag-of-words feature extraction method, with an accuracy of 15.4%, although this is hypothesized to be due to dependence between author and topic. With content-free features, SVM was found to be the best learning algorithm, in contrast to previous findings [8], probably due to the small size of the data set used here. Furthermore, 2-Nearest-Neighbor was found to perform poorly when used with content-free features, again in contrast to previous findings [8]; the reason for this discrepancy is still unknown. Overall, de-pseudonymization of Reddit comments using stylometry-based methods was determined to be practical within sub-communities.

1 Introduction

Outside of "walled gardens" like Facebook, people generally use pseudonymous identities when interacting online. While providing some measure of identity protection, using a pseudonym is not the same as being anonymous. In addition to more traditional attacks on pseudonymity, such as convincing or coercing websites or service providers to disclose users' identities, it has

recently been shown that author identification based on writing style (also known as "stylometry") could practically be used to identify blog authors on an Internet-wide scale [8]. This project attempts to apply current state-of-the-art methods of stylometry-based author identification to comments on the popular Internet discussion site Reddit.com [13].

Reddit is an internet community that thrives on pseudonymity. This is partly due to the culture of the site's community, and partly due to users' practice of creating and maintaining so-called "throwaway" accounts which they use to discuss highly controversial and embarrassing material. (All content on the site is posted in user-created, user-moderated "subreddits" over which Reddit, Inc. exercises close to the minimum amount of oversight required by law.) If it were possible to identify users reliably based on writing style in Reddit comments, it would be possible to connect "throwaway" accounts to users' primary accounts, which can be much more strongly died to their real-world identities (perhaps through a shared username). Reliable stylometric de-pseudonymization of users would have a profound affect on the Reddit community, and on other similarly-structured online communities.

However, paired sets of comments for "main" and "throwaway" accounts of the same user are currently unavailable outside a few specific cases [4, 5]. Additionally, while it might be socially beneficial to unmask certain "throwaway" accounts, in the general case it would be unethical to publish the specific results of a successful experiment into stylometrically pairing accounts, because it could potentially have negative effects on the owners of the accounts thus paired. Moreover, the user base of Reddit is so large that collecting enough data to have comments from both "throwaway" and "main" accounts for a substantial number of users, without knowing beforehand where to look, would be beyond the scale of this project.

For these reasons, this project focuses on a different but related problem: matching individual Reddit comments back to the accounts that posted them. This task has no ethical dilemmas associated with it, and it is far easier to evaluate the success of a machine learning algorithm, because the data is labeled. It is also possible to run an informative experiment on a data set of any size. Algorithms that do well at this task could be expected to do relatively well on the throwaway-main matching task, while algorithms that do poorly on this task could be expected to do similarly poorly on that task.

This project evaluated the accuracies of combinations of three different machine-learning classification algorithms (2-Nearest-Neighbor, Naive Bayes, and SVM) and two different feature extraction methods (bag-of-words and

content-free) at the task of matching comments to authors. The data set used consisted of 70,744 author-labeled comments, representing 100 distinct users with at least 100 comments each. Comments were divided into training and test sets, and the most accurate combination of feature extraction method and learning algorithm was Content-Free features with the Naive Bayes algorithm, with an overall generalization accuracy of 9.5% (versus an expected 1% accuracy for uniform random guessing).

2 Related Work

This project is partly intended to reproduce the results of the 2012 paper "On the feasibility of Internet-scale author identification" by Narayanan *et al.* [8]. In that study, researchers attempted to match sets of three blog posts back to their source blog, by extracting stylometric (here referred to as "Content-Free") features from the text of posts and employing a variety of machine learning classification algorithms. Overall, the experiment was able to achieve a maximum post-to-blog accuracy rate of almost 20%. The "Internet-scale" assertion comes from the very large data set employed: there were 100,000 blogs in the experiment, and thus 100,000 potential classes for each set of three posts. Consequently, the 20% accuracy figure is actually quite impressive.

The most interesting element of the Narayanan *et al.* study is its feature selection methodology [8]. The classic solution to text classification problems involves using a bag-of-words feature set (usually combined with a Naive Bayes classifier). However, the bag-of-words feature set is poorly theoretically motivated for author-identification problems when different authors are expected to write about the same topics. For example, two different authors writing about a current news event are likely to produce blog posts with very similar "word bags". Put another way, a post by author A on topic 1, under the bag of words model, would look more similar to a post by author B on topic 1 than a post by author A on topic 2. The qualities that make bag-of-words feature extraction useful for topic determination make it a poor choice for author identification if you want to separate author and topic cleanly. (However, note that, in practice, author and topic may be highly dependent in some data sets.)

Instead of a bag of words, the Narayanan *et al.* study chose to use a set of nearly purely stylometric features, including text length, measures of

vocabulary richness that could be computed from posts, word length and capitalization characteristics, individual character frequencies, a bag of "function words", and edges of syntactic parse trees [8]. These features, it is argued, capture characteristics of an author's writing style that can be expected to vary relatively little across topics or contexts.

The vocabulary richness features used in Narayanan *et al.* deserve some explanation, as their nature is not obvious; these include Yule's K and measures of *hapax legomena* and *dis legomena* [8]. Yule's K is a measure of the breadth of word choice in a piece of text, used primarily due to its length-independence [7]. *Hapax legomena* are words that occur only once in a given text [10], and *dis legomena* are words that occur twice in a given text. Both of these are also popular metrics in stylometric analysis.

The primary contribution of the Narayanan *et al.* study over previous studies [1] is the introduction of features based on the syntactic structure of the text under analysis. All sentences in the blog posts under analysis were parsed with the Stanford parser to create syntactic trees, and the frequencies of (parent, child) relationships between syntactic projections (excluding the minimal projections, which were the words themselves) were calculated and used as features [6, 8]. These features should help to reveal syntactic structures that some authors might use heavily and others might not use at all, and which might be expected to be independent of context or topic. In practice, these features were found to provide a boost of 2 percentage points in post-to-blog matching accuracy [8].

The claim of 20% author identification accuracy from a pool of 100,000 authors from the Narayanan *et al.* study is impressive, but it is important to put it in context. The sample texts used for author identification were reasonably substantial; groups of three blog posts were used as test points, and blog posts, depending on the blog, can run into the hundreds or even thousands of words. Moreover, in the post-to-blog matching task, topic was almost certainly conflated with author to some extent. Many of the 100,000 blogs almost certainly shared topics, some blogs were probably general-interest blogs writing on many topics, and the feature-selection method used effectively stripped out most content information. However, some content information may have leaked through: blogs about programming, for example, likely had a higher frequency of special characters than other blogs, while blogs on certain topics (perhaps politics or finance) could have had distinctive signatures in the bag of "function words" features, which included words such as "majority", "minority", "quarter", and "saving" [8].

3 Methodology

The present study adapts many of the techniques used by Narayanan *et al.* to the domain of comments on the Internet discussion site Reddit.com. Much like the posts-to-blog task, this study attempts to match single comments to pseudonymous commenters [8].

3.1 Data

The data used in this experiment came from user comments on the web site Reddit.com [13]. A complete list of the user names of the comment authors (some of which were unfortunately quite vulgar) can be found in the Acknowledgements section.

Data Set Characteristics The original data set for the project project consisted of 237,889 comments from 514 users from the comments sections of Reddit posts from the "front page" "subreddits" of Reddit.com, downloaded in several batches over the course of two days in November 2012 [13]. However, due to run-time considerations, only a subset of this data could be analyzed: 70,744 comments from 100 users. The analyzed data set size (before filtering out users with fewer than 100 comments) is 14 megabytes, stored as Python tuples of (user name, comment, creation time) serialized using the `pickle` module. Comment text is in the form of Unicode strings with inline formatting, known as "Reddit Markdown". The average length of analyzed comments was approximately 149 characters—barely longer than a Twitter post. Consequently, absolute accuracy levels approaching those of the Narayanan *et al.* study, which operated on sets of three blog posts, were not expected.

Data Collection The data was downloaded using the Python Reddit API Wrapper (`praw`) Python module [3]. For each post in the site's front-page post listing, the list of users who commented on that post was obtained. For each unique user, up to 1000 of their comments, starting with the most recent, were downloaded. This process was continued until a certain number of users with over a certain number of comments was reached, or until an error occurred and no more data could be obtained.

Data Subdivision For each user, each of the downloaded comments was allocated with 50% probability to either the training set or the test set. Each combination of feature extraction method and learning algorithm was trained on the same training set, and then tested on the test set. No hyperparameter tuning was attempted, and no validation set was used.

3.2 Feature Extraction

Comments as downloaded consisted of Unicode strings; very few machine learning algorithms are able to operate on Unicode strings. Therefore, useful features had to be extracted from the comment text.

Bag-of-Words If there is a "classic" feature set for text classification, it is the bag-of-words. In this study, a bag-of-words feature extraction method was employed for comparison against the more complex content-free methods. Specifically, each comment in the training set was tokenized into words (without stemming), and a feature was created for each word. The feature value was the number of instances of that word in the comment. Note that no features were created for words in the test set that did not appear in the training set.

Content-Free To compare against bag-of-words, a "Content-Free" feature extraction method was also implemented, following the feature extraction protocol of Narayanan *et al.* as closely as possible [8]. However, some of the vocabulary richness features of that study—specifically, Yule's K and the features described only by "etc."—were too poorly documented to implement in the present study. Additionally, due to computational constraints, the syntactic structure features had to be replaced with similar, easier-to-compute features. The features actually used were:

- The number of words in the comment.
- The number of characters in the comment.
- Yule's K. As there has been significant confusion in the literature as to how Yule's K is actually calculated [7], and no available academic source that explains its notation sufficiently well for implementation in software (the original description by Yule having been published in

1944 [15]), Yule's K had to be calculated by the method described on the blog "A Geek with a Hat" [14].

- The number of words appearing exactly n times in the comment, for $1 \leq n \leq 10$.
- The number of words with exactly n characters, for $1 \leq n \leq 20$.
- The number of words in all upper-case, all lower-case, with the first letter capitalized, with the first letter and some subsequent nonadjacent, non-final letters capitalized (CamelCase), and with any other pattern of capitalization.
- The number of instances of each letter A–Z, ignoring case.
- The number of instances of each digit 0-9
- The number of instances of each other ASCII-encodeable character present in the training set.
- The number of instances of each word on the list of 297 "function words" from Narayanan *et al.* [8].
- The number of words in each part-of-speech category. Producing full parse trees as in Narayanan *et al.* in a computationally feasible way proved to be a prohibitively difficult engineering problem, so part-of-speech tagging, a computationally simpler task, was substituted for parsing. As the most informative parse tree features of Narayanan *et al.* involved part of speech nodes, this substitution is not predicted to have too much of an impact on overall prediction accuracy [8].

3.3 Feature Normalization

First, each feature was rescaled to have variance one. Other feature rescaling methods would have been theoretically more appropriate, but could not be implemented in combination with the sparse matrix data structures that the large size of the bag-of-words feature vectors required [8]. Next, each feature vector was re-scaled to unit length.

3.4 Classification Algorithms

This experiment used three classification algorithms: Naive Bayes (using a multinomial distribution), 2-Nearest-Neighbor, and SVM. The Naive Bayes algorithm was included because it is the "classic" text classification algorithm (especially when paired with bag-of-words features). A multinomial distribution was chosen because it has theoretically nice properties for bag-of-words-based text classification, and because the Naive Bayes implementation available that used Gaussian distributions could not handle sparse matrix data. The 2-Nearest-Neighbor classifier was chosen because of its high performance on the comment-to-blog matching task, and its easy off-the-shelf availability [8, 11]. The SVM classifier (more formally, a linear Support Vector Classifier) was chosen primarily due to its popularity for other problems; it did not perform particularly well on the posts-to-blog matching task. However, as there are more features than classes in the present experiment, SVM might be expected to perform better than it did in the Narayanan *et al.* study [8].

3.5 Computational Resources

The data downloading was carried out from a personal laptop over an institutional Internet connection. The primary limit on the amount of data downloaded was the API request throttling policies of Reddit [3]. Nevertheless, data download rate was not the limiting factor in the size of the analysis performed.

The computational analysis was carried out on `campusrocks-2-1`, one of the nodes of the CampusRocks cluster at the University of California, Santa Cruz. Feature extraction was parallelized across the system's 48 CPU cores, and used about 20 gigabytes of the machine's approximately 200 gigabyte memory space. The speed of this machine at executing off-the-shelf machine-learning algorithms (which did not appear to be parallelized across more than one core) was the limiting factor in the size of the experiment.

3.6 Software Tools

This project relied heavily on Python and on a number of python libraries of various levels of standardization and quality. Additionally, an attempt was made to utilize the Stanford parser, as in Narayanan *et al.*, to parse the

syntactic structure of comments and derive syntax-based features [6, 8].

Python 2.7 One of the most challenging aspects of this project involved getting a functional and reasonably up-to-date copy of the widely-used Python programming environment running on `campusrocks-2-1`. The system's default Python was version 2.4, first released in 2004 [12]. The system had an installation of Python 2.7, but this had not been configured correctly, and was unable to download and install the necessary Python modules to run the analysis script. Consequently, before any serious data processing could be done, a new copy of Python 2.7 had to be installed and configured under an unprivileged user account. Then the required modules (and their dependencies, reaching all the way down to the Fortran libraries BLAS and LAPACK) had to be downloaded, compiled, and installed. All of this work cut into the time available for actually running computational analysis, and reduced the size of the data set that could be processed.

Python Reddit API Wrapper (PRAW) Downloading comments from Reddit.com was accomplished via PRAW, the Python Reddit API Wrapper. This library abstracted away all of the details of synchronous communication with the site's remote servers, and made downloading large amounts of data in a machine-readable form fairly trivial [3]. It should be noted that the existence of such a web API and an associated easy-to-use Python wrapper makes the programmatic de-pseudonymization of Reddit users significantly easier.

Natural Language Toolkit (NLTK) The Natural Language Toolkit, or NLTK, is a Python module for analyzing and processing English text. It contains off-the-shelf components for tokenizing text into words and for part-of-speech-tagging words, which were used in this analysis, among a number of other capabilities. However, while it does contain implementations for a variety of sentence parsing algorithms, and tools for working with parse trees, it does not include a grammar for English, and consequently cannot be used to parse English text [2]. Additionally, although it contains implementations for several basic text-classification machine learning algorithms, these implementations were not sufficiently robust or general for the purposes of this experiment [9].

scikit-learn Instead of the NLTK classifiers, this experiment used the more robust and versatile classifiers found in the scikit-learn package, which provided off-the-shelf implementations of the Nearest-Neighbor, Naive Bayes, and SVM classification algorithms, as well as various functions for manipulating labeled feature vectors [11]. Although highly useful and efficient, the scikit-learn classifiers still had some notable shortcomings. The most severely limiting was their lack of parallelization; scikit-learn could only utilize $\frac{1}{48}$ of the computing power available to run the experiment. Had these algorithms been efficiently parallelized, it would potentially have allowed for analysis of all of the data collected, instead of just a subset.

The Stanford Parser To compute syntactic-structure-based features, Narayanan *et al.* used the Stanford Parser [6, 8]. The Stanford Parser is a Java application which produces statistically-motivated parse trees for arbitrary English sentences [6]. It is by no means perfect, but it might reasonably be considered state of the art. Unfortunately, it does not have a mature Python interface. In the absence of such an interface, an attempt was made to wrap the Stanford parser executable (which operates on plain text files or streams) for use by the Python analysis script. This attempt failed for a variety of reasons, the most salient being an approximate 2-second startup time which, because of the way the parser handles output buffering, would have had to have been incurred for every comment. Consequently, instead of the Stanford parser, NLTK's integrated part-of-speech tagger was used to create a roughly comparable set of features.

4 Results

The results of the experiment are presented in Table 1. As can be seen there, bag-of-words feature extraction was found to produce higher classification accuracies than content-free feature extraction (up to 15.4% correct, and having the correct user in the top 5 30.0% of the time with a Naive Bayes classifier). Furthermore, the accuracy of the SVM classifier was found to be the highest for the content-free feature extraction method, and that of 2-Nearest-Neighbor the lowest. Both of these results are in conflict with Narayanan *et al.* (although that study did not directly examine bag-of-words feature extraction, dismissing it *a priori* for theoretical reasons) [8].

Learning Algorithm	Metric	Feature Extraction Method	
		Bag-of-Words	Content-Free
2-NN	Accuracy	4.6%	4.2%
	Top 5	11.7%	11.0%
NB	Accuracy	15.4%	7.2%
	Top 5	30.0%	20.9%
SVM	Accuracy	14.9%	10.9%
	Top 5	–	–

Table 1: The accuracies of all combinations of feature selection methods (bag-of-words and content-free) and learning algorithms (2-Nearest-Neighbor, Naive Bayes, and SVM). Accuracy is the portion of comments in the test set correctly paired with the user who posted them. Where made possible by the off-the-shelf classifiers used, the portion of comments for which the correct user was in the top 5 most likely users, as estimated by the classifier, is also presented. Note how bag-of-words features outperform content-free features for every classifier tested. Also note how SVM is the most accurate classifier for both feature extraction methods.

5 Conclusions

The first conclusion that can be drawn from the results of this experiment is that identifying a user from a pool of 100 is often possible given only a single comment. An SVM classifier operating with a bag-of-words feature set can correctly identify the user in more than 1 in 5 cases. This suggests that an SVM/bag-of-words-based attack on Reddit user pseudonymity within the context of one of Reddit's small "subreddits" could be viable with even very limited computing resources.

The high user identification accuracy rates are surprising given that the average length of a comment in the data set analyzed is a mere 149 characters. Assuming that extremely short comments cannot be identified reliably, the correct-identification rate of longer comments must be correspondingly higher.

The fact that the bag-of-words feature selection method unexpectedly outperformed the stylometric content-free feature selection method cannot be explained by the absence of syntactic-structure-based features; such features provide only a modest increase in accuracy [8]. Instead, the high per-

formance of the bag-of-words feature extraction method implies that, in the data collected, user identity and topic are strongly connected. Because comments were downloaded on a per-user rather than per-topic basis (in order to ensure that there were enough comments from each user to make them plausibly identifiable), there is not necessarily a large number of comments in the data set on the same topic by multiple users. Furthermore, as the number of users was only 100 in this experiment, it is plausible that each user had some particular idiosyncratic interest to which they devoted a nontrivial fraction of their comments. This hypothesis is supported by the very high (30.0%) rate of the correct user being in the top 5 users for the comment as identified by Naive Bayes; the algorithm could be identifying a small group of users, all of whom discuss a particular topic, as being likely to produce comments on that topic.

To test the hypothesis that a dependence between user and topic is inflating the accuracy of the bag-of-words feature extraction method, one could carry out a version of this experiment restricted to a single "subreddit" with a well-defined community; this would effectively make user identity conditionally independent of topic by conditioning on topic, while the small number of users in the community would make it easy to obtain a sufficient number of comments per user.

The results obtained here are largely inconsistent with those of Narayanan *et al.* [8]. In addition to the unexpected accuracy of the bag-of-words model, the best algorithm from that study, 2-Nearest-Neighbor, was the worst algorithm in this experiment, while the least accurate algorithm in that study, SVM, was the best here.

The high performance of the SVM algorithm in this experiment is most likely attributable to the small number of classes relative to the number of features used. While 100 classes may seem like a substantial number, it is easy for them to all be linearly separable in a space with more than 100 dimensions. The content-free feature extraction method produced almost 300 features from the function words counts alone, while the bag-of-words method produced thousands of features. The fact that the number of dimensions was greater than the number of classes made the classification problem in this experiment fundamentally different from that in the Narayanan *et al.* study, eliminating "masking" and allowing SVM to perform well [8].

The low performance of 2-Nearest-Neighbor is more difficult to explain. It is potentially attributable to differences in per-feature scaling; Narayanan *et al.* use their own rescaling method, which they call "feature-mean-nonzero"

(involving dividing all feature values by the mean of the nonzero feature values), in conjunction with 2-Nearest-Neighbor, while in this experiment the variance was merely re-scaled to 1, leaving the mean unchanged [8]. While the difference in the treatment of the mean should not matter for nearest-neighbor (as it is distance-based), these two schemes could potentially affect the variances differently. An alternative explanation is that the algorithm that Narayanan *et al.* refer to throughout as "NN2" is actually a centroid-based modification of the nearest-neighbor algorithm, while that used here is "stock" k-Nearest Neighbor with $k = 2$ [8]. On a related note, the very idea of a 2-Nearest-Neighbor classifier for discrete classification may be somewhat flawed—if the classes of the two nearest neighbors do not match, what would the classifier return? This question is not addressed by Narayanan *et al.* [8].

In conclusion, the scale of this experiment is too small to effectively support or debunk the results of Narayanan *et al.*; unless the number of features is, as in that experiment, smaller than the number of classes, the relative performances of classification algorithms in the two different experiments are simply not comparable [8]. However, this experiment has shown that author identification on the basis of single Reddit comments, despite their low information content, is feasible, at least in communities smaller than the number of useful stylometric features, or among commenters that have distinctive preferred topics of conversation.

6 Acknowledgements

The author would like to thank the developers of all of the Python modules used in this experiment, without whom the software development aspect of this experiment would have been several orders of magnitude more difficult. The author would also like to thank the Baskin School of Engineering cluster support staff for helping to solve cluster configuration problems on very short notice. Furthermore, the author would like to thank the administrators at Reddit.com for making their valuable community data available to the public through a convenient API.

Although in some cases the author would not like to do so, the author is obligated to thank the following Reddit users for their comment data, without which this project would not have been possible: skadferlyfe, DancingWithKafka, MrsJetson, skaterape, MakeEmSayUhhh, CarolineTurpentine, emohipster, postsinsensitively, PSIStarstormOmega, Dieck_Pwns_All, itsasil-

lyplace, Jinxy_Minx, ShootTheMoon, savant9, thestairway67, parkman, Jormungandrobot, rfp_drew, stringdom, crackjoy, Thomsenite, Uglies_Bumped, Zydrunas, kaiden333, AudibleKnight, asshatnowhere, Pryer, Boomanchu, Shardwing, radbrad7, Dark_Matrix, smellyorange, miseryisnotdead, real_nice_guy, Doctor_Loggins, Imtheone457, TheCollective01, MananWho, Karnas, buttabeens, lethargicwalrus, longcoolwoman, DoorLord, Anonymous3542, dubyat, mateogg, livefreeordont, n8k99, You_Better_Smile, JCJen7, NiceSuit, 1Down, ocarinaofrhyme, mocodity, NothingtodohereOGW, TThor, dingofarmer2004, SanchoDeLaRuse, Sodfarm, Lettuce_Get_Weird, admdelta, guardiandevil, maruraba, 4nz, PixelMagic, ricktencity, sapienshane, tq92, TehGogglesDoNothing, VidCrafter, MortFlesh, Maester_May, FlipStik, MrCheeze, wilu, Engineer3227, Keyserchief, ELEMENTALITYNES, nuxenolith, ISwearThisIsOriginal, tesla3327, PerilPhoSho, xarlev, CountGrasshopper, titan413, lesleh, NigWantsKFC, barnesandnobles, Sleepy_One, tbandit, captgrizzlybear, SockofBadKarma, FUCKED-WITH-A-KNIFE, Loumen, Biased_Dumbledore, Rphenom, letspopbottles, Viper007Bond, SmartViking, and FadedPoste. In many of your cases, the author hopes to be able to de-pseudonymize you soon, so that the author may report your highly inappropriate choice of pseudonym to your parents or guardians.

7 Bibliography

References

[1] A. Abbasi and H. Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems*, 26(2):7, 2008.

[2] S. Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[3] B. Boe. Python Reddit API wrapper 1.0 documentation. `http://python-reddit-api-wrapper.readthedocs.org/en/latest/praw.html`, Nov. 2012.

[4] A. Chen. Unmasking Reddit's violentacrez, the biggest troll on the web. `http://gawker.com/5950981/`

unmasking-reddits-violentacrez-the-biggest-troll-on-the-web, Oct 12 2012.

[5] D. Fitzpatrick and D. Griffin. Man behind 'Jailbait' posts exposed, loses job. http://www.cnn.com/2012/10/18/us/internet-troll-apology/index.html, Oct 19 2012.

[6] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430. Association for Computational Linguistics, 2003.

[7] A. Miranda-García and J. Calle-Martín. Yule's characteristic K revisited. *Language resources and evaluation*, 39(4):287–294, 2005.

[8] A. Narayanan, H. Paskov, N. Gong, J. Bethencourt, E. Stefanov, E. Shin, and D. Song. On the feasibility of Internet-scale author identification. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 300–314. IEEE, 2012.

[9] NLTK Project. classify package. http://nltk.org/api/nltk.classify.html, Nov. 2012.

[10] Oxford University Press, editor. *Oxford English Dictionary*. Dec 10 2012.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] Python Software Foundation. Python 2.4. http://www.python.org/download/releases/2.4/, Nov 2004.

[13] reddit inc. reddit: the front page of the internet. http://www.reddit.com/, Nov. 2012.

[14] S. Teller. Measuring vocabulary richness with python. http://swizec.com/blog/measuring-vocabulary-richness-with-python/swizec/2528, Sep 28 2011.

[15] G. U. Yule. *The statistical study of literary vocabulary.* Cambridge University Press, 1944.