Authorship attribution

1 Introduction

If you want to publish an anonymous or pseudonymous text, what can you do? Before Internet, you could hide your handwriting and make sure you don't leave fingerprints; then just deliver your letter to a postbox and make sure no-one sees you. On the Internet, you could e.g. post your comments using Tor (Dingledine *et al.* , 2004), to prevent anyone from knowing your IP address. [1]

Even if these privacy measures worked perfectly, they aren't enough to protect your anonymity. The content of the message is among the harder things to hide, especially if you write pseudonymously on a regular basis. In this paper, we introduce the problem of *authorship attribution*: how does it work, what can be identified—tweets, source code, for instance—and if, or how, it can be circumvented.

2 Stylometry

2.1 Linguistic problem

Identifying an author is an old problem: criminal investigations, controversial political documents, novels written under a pen name, and anything in between. Brennan *et al.* (2012) mention early approaches such as average word length, vocabulary usage, sentence structure, and function words. In an early study, Mosteller & Wallace (1963) analyse the authorship of The Federalist Papers, and point out seemingly small things, such as one suspected author preferring the word *while*, and another preferring *whilst*. The linguistic theory behind these assumptions is that language is a set of choices, and people tend to be consistent in their choices. Some choices are due to dialectal or social factors, and people use different style in different contexts (instant messaging vs. scholarly essay), but this leaves still a lot of mostly idiosyncratic variation. As explained by Patrick Juola, a computational linguist specialised in stylometry[2]:

> If you ask yourself where the salad fork is relative to the plate, you quickly realize that it's usually to the left of the plate. Or is it? It's just as likely to be "on" the left of the plate, "at" the left of the plate, or perhaps "to" the left SIDE of the plate. Same fork, same position, and at least four different choices for how to describe it, none of which correspond to any sociolinguistic or cognitive variable with which I'm familiar.

If this theory is correct, then it means that authorship attribution should be possible even in different genres of text. Conversely, if we can pinpoint the variation to concrete words and syntactic patterns, then it should be possible to hide one's authorship, either by mimicking someone else, or opting to a random variant, or even possibly constructing a neutral variant of language.

[1] Of course, you can still be identified from pretty much anything, like browser extensions or errors in your computer clock (Murdoch, 2006).

[2] http://languagelog.ldc.upenn.edu/nll/?p=5315

2.2 Machine learning problem

Back when Mosteller & Wallace (1963) did their experiments, it took 3 years to analyse the data. Nowadays we can process larger amounts of text, and use more, and more complex, measures. The paradigm of *machine learning* is helpful: we don't know beforehand which features are discriminative, but we can try a huge amount of features, most of which would be counterintuitive for humans to think of, and impractical (or impossible) to detect.

Classification We can pose authorship attribution as a classification problem or as a clustering problem. Classification means that we have a set of candidate authors and an unknown text, which we want to attribute to one of the authors. Clustering is more suitable approach, if we have a large set of texts, such as forum posts by different usernames, and we want to find out if some of the users are actually the same person.The classification task can be two-class or multi-class: the former being "was it the suspect or someone else", and the latter "which one out of the suspect pool". As intuitively expected, smaller suspect pool leads to larger accuracy. However, even with an extremely large world, the methods do better than random guessing: Abbasi & Chen (2008) report 30 % accuracy for a set of 100,000 authors.

Features In order to measure similarities between texts, the texts are reduced into *feature vectors*. To give a small example, imagine we have the task of language recognition: given a piece of text, output the most likely label in {*English, French, Swedish*}. Comparing against a huge dictionary (including all inflected forms) would be slow, and not work for unknown words. Instead, we assume that each language has a characteristic *letter distribution*. Hence, we can reduce a text into a vector of letter counts. For instance, the text "the cat sat on the mat" corresponds to the vector `[#a=3, #b=0, #c=1,#d=0, #e=2 ... #t=5 ... #ö=0]`. Then we can compare this unknown vector to the vectors that contain average letter distributions of the candidate languages, by simply calculating the angle between two vectors.

There are, of course, a variety of features and more complex similarity metrics, but the principle is the same: reduce something as abstract as "being in English" into something that can be computed efficiently and unambiguously.

We return to the question of feature selection when presenting each paper.

3 Adversarial stylometry

So far we have dealt with honest data: the authors are not deliberately trying to hide their identities. Some of these measures are easily detectable by humans too, for instance synonyms and syntactical structure: if someone tends to write extremely long sentences and use the word *kerfuffle* a lot, people will likely suspect them as an author of similarly written anonymous letter. Brennan *et al.* (2012) ask the question: can we modify our writing and fool stylometric tools? They experiment with three strategies, which are presented in the following sections: obfuscation, imitation and machine translation.

3.1 Obfuscation and imitation

There were 45 test subjects, who provided 6500-word samples of formal text in their usual writing style. In the first experiment, they were instructed to write a 500-word text, delib-

erately trying to hide their style. In the second one, they were instructed to write in the style of Cormac McCarthy. The topics were chosen by the experimenters, and same for all subjects: describing their neighbourhood and typical daily routines. The test subjects were mostly college educated, but none was a professional writer.

The test was successful: the adversarial passages fooled all tested classifier methods, which performed well for conventional methods. The accuracy dropped from 80-90 % (depending on the suspect pool size) to below 20 %; in the obfuscation test, around random change, and in the imitation scenario, worse than random.

The subjects did not receive any particular orders for the obfuscation task, and it was conducted before the imitation task. The most common strategy was to write shorter sentences, avoid specialised vocabulary or "fancy words"; in general, the Flesh-Kincaid grade level was lower in the obfuscated passages.

3.2 Machine translation

Writing in someone else's style is cognitively taxing, and hard to keep up, if one plans to write multiple documents. Hence, Brennan *et al.* (2012) investigated machine translation as an option: the author writes a text in their normal style, and then does a machine translation round trip. In the paper, the authors try one-step translation English-German-English and English-Japanese-English, and two-step translation, English-German-Japanese-English.

The experiments lowered the chances for identification, a but in practice the results were not usable. Complex sentences come back garbled, which in a sense works for obfuscation but not for communication. Simple sentences stay too faithful, and not obfuscated enough. Only a small number sentences worked as expected.

The experiments were run in 2012, on two freely available programs, Google Translate and Bing. Both systems at the time were based on statistical methods. In short, a statistical MT system is trained on large corpora of bilingual aligned data: initially, the system gets the aligned sentences "the cat sat on the mat" and "de kat zat op de mat", and the system will have to infer on its own the smaller chunks: that English "cat" corresponds to Dutch "kat" with a certain probability in a certain context. For some words the correspondence is very clear, like cats and mats, but for some words it depends a lot on the context: for instance, the definite article is always "the" in English, but Dutch has two genders, and thus the translation may be either "de" or "het", depending which word follows.

At the time of writing, statistical MT was the only feasible method for wide-coverage MT. Now, 5 years later, neural networks are more widely used in machine translation, and it is possible that they could offer some improvements. An example is the notion of context: statistical MT has a limited window of concrete words, directly before or after the word(s) to be translated. In contrast, there are variants of neural MT which bypass this hard limit of surrounding words.

Purely rule-based methods predate both of these methods by decades—simple dictionary-based MT is old as computer itself, and has been successful in tasks that require high precision, but only need to cover a small domain. Given that rule-based MT is even more deterministic than statistical, and an ideal machine translation round-trip should be the identity function, we can safely rule out rule-based MT as an obfuscating tool.

M. Brennan et al.

Table II. Writeprints Static Feature Set

Group	Category	No. of Features	Description
Lexical	Word level	3	Total words, average word length, number of short words
	Character level	3	Total char, percentage of digits, percentage of uppercase letters
	Special Character	21	Occurrence of special characters
	Letters	26	Letter frequency
	Digit	10	Digit frequency (e.g., 1,2,3)
	Character bigram	39	Percentage of common bigrams
	Character trigram	20	Percentage of common trigrams
	Vocabulary Richness	2	Ratio of hapax legomena and dis legomena
Syntactic	Function Words	403	frequency of function words
	POS tags	22	Frequency of Parts of speech tag
	Punctuation	8	Frequency and percentage of colon, semicolon, qmark, period, exclamation, comma

The Writeprints-Static feature set, adopted from the Writeprints approach [Abbasi and Chen 2008].

Figure 1: Writeprints feature set

3.3 Features and classifiers

The study experiments with different classifier methods, and feature selection. The full details are presented in the paper, here we only recap the most successful method: Support Vector Machine classifier with a customised Writeprints (Abbasi & Chen, 2008) feature set. The idea behind Writeprints is that each author may have a different feature vector. The full set includes tens of thousands features, most of which are dependent on the documents to classify, such as common misspellings. For this experiment, the authors used the subset of 557 features that were all static.

4 Frontier paper I: Cross-domain authorship attribution

Does writing style stay consistent even in different genres? If we only have scientific papers of an author, could we possible recognise their tweets?

Overdorf & Greenstadt (2016) report that baseline is very low: conventional stylometric methods, which achieve 85 % accuracy for in-domain text, drop to a mere 35 % when trained on one domain and tested on another. With some changes in the feature set and classification algorithm, they are able to achieve 70 % accuracy. Cross-domain authorship attribution is limited to closed suspect pool: whereas previous studies have tested 50-100 authors with moderate success, here the method loses a lot of accuracy after 10 authors.

4.1 Technical details

The study used a subset of Writeprints set; A naive approach of using only function words does not improve the results from the baseline 35 %.

Distortion If we have text from known authors in both source and target domain, we can model the distortion. For a human, this would be like a rule of thumb that "in Twitter,

people are using abbreviations", and hence one should expect that an academic, who never uses abbreviations in their scientific papers, may still do so in a tweet; so a human can learn to ignore the intuition "this cannot be Professor Smith" when reading a tweet "where r u ppl?". For a machine learning method, distortion is defined as a function of feature vectors. We take feature vectors of the same author in different genres, and apply an appropriate distance measure: Overdorf & Greenstadt (2016) use Euclidean distance between the feature vectors.

Finding the best combinations Better results are achieved by Recursive Feature Elimination: running experiments repeatedly on a development corpus, to find the combination of features that does the best in classifying the known cases. However, the authors report that the most discriminating features for in-domain task are often among the most distorted, and hence not good for cross-domain analysis. Applying distortion to the unknown vectors results only in worse results. Hence, the authors conclude that feature selection alone is not an effective method for cross-domain authorship attribution.

Changing the classifier The authors report better results when changing the classifier: training with samples of both domains. Then, the classifier result will reflect more accurately what separates the unknown sample from authors in both source and target. This means that features that discriminate well regardless of genre, are weighted higher. As a difference to specifying the feature set beforehand, this method is more flexible, adapting to different instances.

5 Frontier paper II: Source code stylometry

An experiment by Caliskan-Islam *et al.* (2015) reveals that program code is also easy to identify. Using data from Google Code Jam, where all the programmers completed the same tasks, they are able to identify the author with 98 % accuracy for a set of 250 programmers, and 94 % for 1,600 programmers. They had data from multiple years for 25 programmers; training with old code and testing with new code resulted in similar accuracy, 98.4 %. Of course, the sample size was smaller, so it is possible that the results would drop for a larger pool. Furthermore, the scenario was easier than reality, where the available code could be in a different programming language, and the tasks are more varied.

5.1 Technical details

The authors used a random forest classifier. Random forests consist of decision trees, each using slightly different features and values: one tree really values some features like whether the programmer starts a {-bracket on a newline, and may split the target set into "either-Alice-or-Bob" and "someone-else" (not Alice or Bob) based on those newlines. Another looks at some other features, and makes a split "either-Alice-or-Carol" and "someone-else" based on, say, the programmer's tendency to use nested list comprehensions. Individually these decision trees would make bad decisions, but during the training, they are learning the best features and weights, and together they reach high accuracy.

The authors identify 928 features that are most informative. They can be divided in three classes:

1. Layout features, e.g.

- whitespace
- placement of {}

2. Lexical features; e.g.
   - term frequency of word n-grams
   - average line length
   - average number of parameters per function
3. Syntactic features; e.g.
   - depth, nesting
   - term frequency of AST node n-grams

Layout features contribute around 1 %, lexical and syntactic contribute the remaining 44–55 %. The full set of features can be found in the paper. In contrast to previous work in code stylometry, this study uses syntactic features extensively. For incomplete code snippets, they use a fuzzy parser to obtain the AST.

5.2 Obfuscation

Due to the heavy use of syntactic features, simple obfuscators don't work well to hide the author: they only make the code unreadable by humans. This would be unusable for open-source scenarios, because the code is effectively unmaintainable. More sophisticated obfuscator (Tigress) succeeds better, but makes the program 9 times slower, so this is not a viable alternative either. The authors suggest future research on creating a neutral programming style.

References

Abbasi, Ahmed, & Chen, Hsinchun. 2008. Writeprints: A Stylometric Approach to Identity-level Identification and Similarity Detection in Cyberspace. *ACM Trans. Inf. Syst.*, **26**(2), 7:1–7:29.

Brennan, Michael, Afroz, Sadia, & Greenstadt, Rachel. 2012. Adversarial Stylometry: Circumventing Authorship Recognition to Preserve Privacy and Anonymity. *In: ACM Transactions on Information and System Security.*

Caliskan-Islam, Aylin, Harang, Richard, Liu, Andrew, Narayanan, Arvind, Voss, Clare, Yamaguchi, Fabian, & Greenstadt, Rachel. 2015. De-anonymizing Programmers via Code Stylometry. *Pages 255–270 of: Proceedings of the 24th USENIX Conference on Security Symposium.* SEC'15. Berkeley, CA, USA: USENIX Association.

Dingledine, Roger, Mathewson, Nick, & Syverson, Paul. 2004 (August). Tor: The Second-Generation Onion Router. *In: Proceedings of the 13th USENIX Security Symposium.*

Mosteller, Frederick, & Wallace, David L. 1963. Inference in an Authorship Problem. *Journal of the American Statistical Association*, **58**(302), 275–309.

Murdoch, Steven J. 2006. Hot or not: Revealing hidden services by their clock skew. *Pages 27–36 of: In 13th ACM Conference on Computer and Communications Security (CCS 2006*. ACM Press.

Overdorf, Rebekah, & Greenstadt, Rachel. 2016. Blogs, Twitter Feeds, and Reddit Comments: Cross-domain Authorship Attribution. *PoPETs*, **2016**(3), 155–171.