Stylometry-based Fraud and Plagiarism Detection for Learning at Scale

Markus Krause, markus@hci.uni-hannover.de, Leibniz University

Fraud detection in free and natural text submissions is a major challenge for educators in general. It is even more challenging to detect plagiarism at scale and in online classes such as Massive Open Online Courses. In this paper, we introduce a novel method that analyses the writing style of an author (stylometry) to identify plagiarism. We will show that our system scales to thousands of submissions and students. For a given test set of ~4000 users our algorithm shows F-scores of over 90%.

1 Introduction

Fraud, cheat, and plagiarism detection are major challenges for learning at scale. Verifying that a student solved an assignment alone is extremely hard to verify in an online setting. Even in offline courses with a couple of hundred students, detecting plagiarism or cheating is nearly impossible for a teacher. Various attempts have been proposed to solve this issue (Meuschke, 2013). Intrinsic plagiarism detection is a promising method for large-scale online courses. Intrinsic plagiarism detection uses stylometry (analysis of literary style) to identify stylistically different segments in a text. Features used for stylometry are sentence length, vocabulary richness, frequencies of words, or word lengths. With carefully chosen features, stylometry is robust even against automated imitation attempts as automatically altering the grammatical structure of a sentence without changing the meaning of the text is challenging (Brennan, Afroz, & Greenstadt, 2012; Krause, 2014). Narayanan and Paskov also demonstrated that stylometry is scalable. They reliably identify authors in a large corpus with texts from 100,000 individuals (Narayanan & Paskov, 2012). Monaco et al. (Monaco, Stewart, Cha, & Tappert, 2013) gives an overview of different identification methods using stylometry.

In this paper, we propose a new method using stylometry for plagiarism detection. We illustrate the feasibility of our method and demonstrate that our approach scales to thousands of authors. Our approach uses a relatively small feature set (164 features) compared to methods such as Writeprints (>30,000 features) (Abbasi & Chen, 2008). In contrast to other approaches, we use an input resampling and smoothing method on the training, testing, and evaluation data and train a classifier for each author. We also use a standard SVM in contrast to most other approaches that use a single class SVM (Abbasi & Chen, 2008). With this method, the SVM can not only learn positive instances but also negative. We hypothesize that:
Negative instances in the training set increase the quality of fraud prediction.

To illustrate the performance of our method we predict if a text was written by a given author or not comparing the writing style of the questionable text to a known sample. We report two main units F-score and Cohen’s Kappa. The F-score is defined as the harmonic mean of precision and recall. Cohen’s Kappa is a measure for inter rater agreement (Cohen, 1960) and also useful in estimating classifier performance. Measuring the disagreement between the algorithms predictions and the expected classes.

We test our method on a corpus build from 17,000 blogs. The corpus was initially composed and published by Schler et al. (Schler, Koppel, Argamon, & Pennebaker, 2005). We selected all blogs with at least 35,000 characters (approx. 3100 words per author).

2 Feature Extraction

For our experiments, we extracted a set of features from each blog in our corpus. Many approaches use features that an algorithm can easily alter, for instance digits. An algorithm can easily detect fractional numbers and add additional numbers to better resemble another author e.g. altering 0.98 to 0.982. This does not change the meaning of the text and a reviewer would not be able to recognize such changes. Similar approaches also work for whitespaces such as line breaks, tabs, and space. Besides omitting certain features, we also expanded others. An often-used feature is average word length. Instead of the average length, we use word length frequencies. Furthermore, we added new features not yet explored. We describe individual feature sets below. They sum up to 164 individual features.

Character Frequency (48 features)
The relative frequency of individual characters. This feature set contains the relative frequencies of a-z and A-Z.

Word Length Frequency (20 features)
The relative frequency of word length. In some rare cases the part of speech tagger was not able to filter certain artifacts e.g. long numbers, some e-mail addresses (without the @ sign). This results in particular long words. To filter such elements we only use words of up to 20 characters.

Sentence Length Frequency (35 features)
The relative frequency of sentence length. Similar to the word length feature we filter out overly long sentences longer than 35 words.

Part of Speech Tag Frequency (35 features)
For this feature set we use the Penn Treebank part of speech tag set. We use the Natural Language Toolkit (NLTK [2]) python library to extract these tags from a corpus. We calculate the relative frequency of each tag.
Word Specificity Frequency (20 features)

The specificity of words used by an author is a discriminating feature and a relevant predictor in other Natural Language Tasks (Kilian, Krause, Runge, & Smeddinck, 2012; Krause, 2013). However, to our knowledge this feature have not been used for stylometry yet. To estimate the specificity of a word we use wordnet (Miller, 1995). For each word, we predict the lemma of the word and its part of speech. With the lemma and the part of speech, we retrieve all relevant synsets. The algorithm calculates the distance between each synset and the root node of wordnet. We define specificity as the average depth of these synsets rounded to the nearest integer. The algorithm calculates the relative frequency of each depth. The depth is limited to 20 as higher values tend to be extremely rare.

3 Method

To represent the diversity of an authors’ writing we resample the input data. We merge all documents of an author and split the resulting text into sentences. Each word in a sentence is annotated with its part of speech (POS). To generate POS we use the Penn-Treebank tagger from the NLTK library. From these annotated sentences, we calculate a feature vector for each sentence. So that each author has a corresponding set of vectors. We split each set into three equally sized subsets a training set, a test set, and an evaluation set.

Each set is further processed. From each set, we randomly select $n$ sentence vectors to have equal numbers of vectors in each set of these sets for each author. We select $b$ bootstrap samples from the training set. A bootstrap sample is drawn by randomly selecting a vector from a set and repeating this as many times as vectors in the set without removing the picked vector from the set. As a final step, we average over all picked vectors to create a single bootstrap sample.

To train the support vector machine for an author we take the $b$ bootstrap samples of an author as positive examples. We then select $i$ other authors randomly from the initial set of ~4000 authors. These authors are called impostors. We again randomly select $\frac{b}{i}$ bootstrap samples from the training sets of these $i$ authors for negative examples.

As we use the described bootstrap method to generate samples, we can use any number for $b$ reasonably smaller than the total number of possible permutations of sentence vectors per author.

For the experiment we used $b=400$, $i=100$, and $n=100$. We generated 400 bootstrap samples for each author and 4 bootstrap samples for each of the impostors, therefore, training the SVM with 800 bootstrap samples. We determined this number with a series of tests. Larger values for $b$ and $i$ only marginally increased classification accuracy. Higher values for $n$ increased classifier performance but seem to be unrealistic for course submissions. We repeated this process 1000 times for each author so we had 1000 classifiers for each author. For each trained classifier we randomly selected 100 authors from the initial ~4000. We again refer to these authors as impostors. These impostors are distinct from those we trained the classifier with! We generated 100 bootstrap samples for each impostor and 100 bootstrap samples for the author. All bootstrap samples are drawn from the sentence vectors selected for
evaluation earlier. We used each classifier to predict the class of each of the evaluation bootstrap samples and calculated F-score and Cohen’s Kappa for each classifier.

We also use a traditional method on the same data set as a baseline as used for the Writeprint system (Abbasi & Chen, 2008). For this method, we use tenfold cross-validation by splitting author texts into 15 parts (5 for training, 5 for testing, 5 for evaluation). For example, in fold 1, parts 1–5 are used for training, while parts 6–10 are for testing; in fold 2, in parts 2–6 are used for training while parts 1 and 7–10, are for testing. From all possible permutations we randomly selected 1000 for each author. We trained a single class SVM with the 5 training samples. To calculate F-score and Kappa we used the evaluation samples of the author and the evaluation samples of a random set of 100 impostors. For both methods, we used the testing samples to estimate optimal parameters for the SVMs. This is a crucial step, as SVMs require parameter tuning to work effectively. To estimate optimal parameters we used grid search on a subsample of authors not used for testing or evaluating.

**Results**

We found our proposed method to be effective in detecting the authorship with a mean F-score of 0.91 and a 95% Confidence Interval of the mean of [0.89, 0.92]. The average Cohen’s Kappa for our method is 0.8 with a 95% CI [0.77, 0.84]. These results are higher than the results from the traditional method with an F-score of 0.65 and a 95% CI [0.56, 0.74] and a Kappa score of 0.46 and a 95% CI of [0.36, 0.56]. Figure 1 illustrates the results of our experiment.

![Figure 1: The binary SVM outperforms a traditional single class SVM. Error bars show the 95% Confidence Interval. We calculated the interval from 10,000 bootstrap samples.](image)
Conclusion

In this paper, we presented a new method using stylometry to validate the authorship of student submissions. We illustrated that our method of input smoothing and resampling allows us to train standard binary SVMs. We also showed that the prediction quality of these binary SVMs trained with our method is higher than the quality of a comparable single class SVM trained with a traditional method. We illustrated that our method performs with high accuracy even for a large data set of more than 4000 authors and reliable predict if a given author wrote a text.

References


