Authorship Attribution Using Writeprints

Ariel Stolerman
Machine Learning (CS613) Final Project
Spring 2012
Drexel University
ams573@cs.drexel.edu

Abstract

Authorship attribution in online domains has become increasingly relevant and important for exposing anonymous identities and cybercrime. In this paper I present an implementation of *Writeprints* [1], a state-of-the-art authorship attribution method. Writeprints is a Karhunen-Loéve-transforms-based technique that uses pattern disruption algorithm with individual author-level feature sets. The implementation is evaluated on the Extended Brennan-Greenstadt Adversarial Corpus [2] in both standard and adversarial settings.

1 Introduction

Stylometry is the application of the study of linguistic style to authorship attribution. The main domain it is used for is written language – identifying an anonymous author of a text by mining it for linguistic features. Authorship attribution has applications across domains such as literature, history and forensic linguistics, with various examples including authenticating works of Shakespeare, identifying the authors of the famous Federalist Papers [3] and analyzing suicide letters.

One of the novel and most effective methods for authorship attribution is Writeprints [1]. In this paper I present an implementation of this method, and integrate it into JStylo [1], an open-source Java-based supervised authorship attribution platform developed at PSAL (The Privacy, Security and Automation lab at Drexel). The platform enables extracting various linguistic features for authorship attribution, and currently supports several Weka [4] classifiers.

2 Background

The field of stylometry has naturally become dominated by machine learning approaches, harnessing computational power to the task to achieve high performance and accuracy. Stylometry challenges can be divided into three main sets: identifying an author from a given set of candidates (supervised learning), classifying a set of anonymous documents into clusters where each belongs to a different author (unsupervised learning) and analyzing text for different author features, such as personality traits, age, gender, number of authors of the questioned document etc. Current research in the field is advancing in three directions:

- Developing stylometry techniques to achieve high accuracy over a varying number of candidate authors, and display better methods of feature selection and cross-domain effectiveness.
- The role of stylometry in privacy and anonymity.
- Adversarial Stylometry - attacking and circumventing stylometry techniques, and countering such attacks.

[1] http://psal.cs.drexel.edu/

2.1 Stylometry, Privacy and Anonymity

Stylometry has an impact over anonymous communication. As users of the internet may want to hide their identity, other than anonymizing their technical identities (e.g. their source IP address), their writing style also has to be taken under consideration. Keeping the privacy and anonymity of internet users may have great value, especially in cases where anonymous communication is essential to exposing crime, government corruption, human rights abuses etc. Therefore in order to keep a private identity, it is necessary to study stylometric techniques and approaches in order to learn how to circumvent such techniques that may be applied on the published communication. Many may think it is sufficient to cover this aspect only for large text samples, but ongoing research demonstrates stylometry abilities that can be applied for short instant messages, twits and similar.

2.2 Approaches

The most common approach to stylometry is supervised analysis of the feature space. Applied in machine learning, this means training a classifier based on a set of known documents of all candidate authors, and applying that classifier to the set of test / unknown documents to determine their author. When approaching a stylometry task, the main configuration that defines the approach is the set of features examined. The domain of possible features is virtually endless (for instance, choosing English letter n-grams alone generates 26^n features), so it is important to select features that best distinguish between authors carefully. Although it seems difficult, decades of research suggest several features that have been proven effective [5]:

- *Function words*: words used to describe a relationship between meaningful words, and are topic-independent. For instance, articles (“the”, “a”, “an”), pronouns (“he”, “she”, “him”) and particles (“if”, “however”, “thus”).
- *Vocabulary features*: the vocabulary richness of a document, or word selection out of a set of synonyms.
- *Syntactic features*: the grammatical structures used by the author to convey his ideas. For instance, various punctuation features such as frequency of different punctuation across the text.

Methods for classification may include neural networks [6, 7, 8], Support Vector Machines (SVM) [9, 10, 11], Bayesian classifiers, decision trees and others. In addition to the above, there are two unique methods designed specifically for stylometry tasks, and have been shown to be promising:

- *The Writeprints method* [1]: developed by Abassi and Chen, this method incorporates a rich set of features, and uses a sliding window and pattern disruption algorithm with individual author-level feature sets. This method is used for both authorship attribution (supervised) and similarity detection (unsupervised). The method is shown to achieve higher accuracy than known before, across tasks with high number of candidate authors.
- *The Synonym-Based method* [12]: developed by Clark and Hannon, this method measures the selection of each word and weighs that measurement by accounting for how common that word is and how many choices the author had.

3 Writeprints

Writeprints is a Karhunen-Loéve-transforms-based method that uses a sliding window and pattern disruption to capture feature usage variance. The method uses individual-author-level feature sets, where each author’s Writeprint is constructed from the author’s key features, attaining greater scalability compared to standard techniques that use a single author-group-level feature set. An author’s Writeprint pattern projects the author’s feature usage variance into a lower-dimension space. All features that an author never used are addressed as pattern disrupters such that the appearance of those features in some anonymous document decreases the similarity between the anonymous identity and the author.

In the next sections the Writeprints approach is detailed, including notes about simplifications and variations in the implementation.

3.1 Implementation Code-Base

The implementation of the Writeprints method presented in this paper is applied as an extension of the JStylo API. As mentioned in the introduction, JStylo is an open-source Java-based authorship attribution platform. The convenient problem set infrastructure and extensive feature extraction tools along with the Weka [4] based data representation (i.e. ARFF files) made JStylo a good code-base to extend.

The Writeprints implementation provides two main Java classes: `WriteprintsAnalyzer extends Analyzer` and `AuthorWPData`. The first, `WriteprintsAnalyzer`, extends the abstract JStylo analyzer, by providing methods targeted for training on a training data and classifying test data or for running cross-validation on training data alone. The second class, `AuthorWPData`, is intended to maintain author's data – basis matrices and writeprint patterns (see details in the following sections). The code is available here[2].

The flow of running an authorship attribution task is as follows:

- Problem set construction: defining the training authors and their documents, along with the anonymous test documents.
- Feature set extraction: extracting features from a pre-defined feature set from all documents.
- Learning: constructing the required data for all training authors and "test" authors (corresponding to each of the test documents).
- Classification: classifying all test documents and attributing the best matching candidate author.

3.2 Feature Set

The Writeprints method uses an extensive feature set with features across different levels of the text: lexical, syntactic, structural, content-specific and idiosyncrasies. In the original paper, the authors experiment with two feature sets: the basic feature-set, consisted of static features, like letter-frequencies, vocabulary richness etc., and the extended feature-set, consisted of both static and dynamic features. Dynamic features are content dependent and may vary from one dataset to another, like bag-of-words, letter n-grams etc. In addition, the authors experimented with two feature-set-types: author-group-level, where all authors share the same feature set, and individual-identity-level, where multiple author-dependent feature-sets were used in one-against-all comparison for each author.

As the feature space is virtually infinite, based on the tested corpus, feature selection heuristic is applied on all n-grams features by measuring information gain [13] and reducing those feature classes to those features with the top information-gain values. Information gain for feature j across a set of classes c is derived as:

$$IG(c, j) = H(c) - H(c|j) \tag{1}$$

Where $H(c)$ is the overall entropy across all author classes and $H(c|j)$ is the conditional entropy for feature j.

3.2.1 Implementation Notes

The authors of the original paper did not provide exact details about all the features used (e.g. vocabulary richness features), so the chosen feature set is only a close approximation of the original. In addition, since we are interested in examining the method some features that address metadata of the documents that is unavailable or irrelevant in the Extended Brennan-Greenstadt Adversarial corpus (like the existence of URLs, font usage, file extensions etc.) are omitted.

[2] https://github.com/psal/JStylo-Anonymouth/tree/jstylo_master/src/edu/drexel/psal/jstylo/analyzers/writeprints

Table 1: Feature set used for the Writeprints method

Group	Category	Quantity	Description
Lexical	Word-level	4	Total words, average word length, frequency of large words (> 4 letters), unique words count
Lexical	Character-level	4	Total characters, % of digits, % of letters, % of uppercase letters
Lexical	Letters	26	Letter occurrence frequencies (e.g. a, b, c)
Lexical	Character-bigrams	< 676 (50)	Letter bigrams (e.g. aa, ab, ac)
Lexical	Character-trigrams	$< 17,576$ (50)	Letter trigrams (e.g. aaa, aab, aac)
Lexical	Digits	10	Digit frequencies (e.g. 0, 1, 2)
Lexical	Digit-bigrams	< 100 (50)	Digit-bigrams (e.g. 00, 01, 12)
Lexical	Digit-trigrams	< 1000 (50)	Digit-trigrams (e.g. 000, 001, 123)
Lexical	Word-length dist.	Varies	Frequency of different word-lengths
Lexical	Vocabulary richness	4	Richness – hapax legomena, dis legomena, Yule's Characteristic K, Simpson's diversity index
Lexical	Special Characters	21	Frequency of special character occurrences (e.g. \$, %, &)
Syntactic	Function words	< 512 (300)	Frequency of function words (e.g. "if", "for", "than")
Syntactic	Punctuation	8	Frequency of punctuation occurrences (e.g. !, ?, :)
Syntactic	POS tags	< 45	Frequency of part-of-speech tags (e.g. NN, VBP, JJ)
Syntactic	POS tag bigrams	Varies (50)	Part-of-speech tag bigrams (e.g. (NN)-(VBP))
Syntactic	POS tag trigrams	Varies (50)	Part-of-speech tag trigrams (e.g. (NN)-(VBP)-(JJ))
Content	Words	Varies (300)	Bag-of-words (e.g. "hello")
Content	Word-bigrams	Varies (50)	Word-bigrams (e.g. "hi there")
Content	Word-trigrams	Varies (50)	Word-trigrams (e.g. "how are you")
Idiosyncratic	Misspellings	$< 5,753$ (50)	Frequency of common misspellings (e.g. "beleive", "thougth")

In addition, for simplification and performance, another heuristic is adopted for reducing the number of features instead of information gain: the feature classes with potential high number of features (e.g. word bigrams) are reduced to the top n used features, across all authors. The n used is 50 in most cases, except for bag-of-words and function words, where n is chosen as 300, due to the importance of those features as shown in previous research. The top-used heuristic was tested on the Brennan-Greenstadt Adversarial Corpus [2] (differs from the **extended** one) which contains documents of 13 authors, and have shown to attain high accuracy for both the Writeprints method and SVM (details of experimental settings and evaluation in section 4). Both heuristics are available in the presented Writeprints implementation, with the top-used as default, due to performance issues.

For simplicity of representation and performance, in the presented implementation only author-group-level feature set was used, i.e. dynamic features take into account all authors. This representation simplifies computations detailed in the following sections. Although The original authors chose individual-identity-level feature sets, which outperformed author-group-level feature set in most of their experiments, it would be much less efficient in both memory and computation. It should also be noted that the comparison of Writeprints with individual-identity-level feature sets in the original paper was not to Writeprints with author-group-level feature set, but to SVM. Thus, it is not clear whether there is a significant difference in accuracy between the two types applied

in Writeprints. As this implementation focuses on achieving high performance, author-group-level feature set was chosen.

The complete feature set used is detailed in table 1. Features with the "$<$" sign in their quantity signify the maximum number of features possible (if no reduction is applied, e.g. at most 676 letter bigrams), and "varies" signifies the quantity corresponds to the content (e.g. bag-of-words). The number in parenthesis signifies the reduction value applied to that feature class (e.g. (300) for bag-of-words).

3.3 Writeprint Creation

The creation part is where lower-dimensional usage variation patterns are created based on the occurrence frequency of the authors' individual features. For all features with usage frequency greater than 0, a sliding window is run over the author's documents. By applying Karhunen-Loéve transforms, feature occurrence vectors for each window are projected to an n-dimensional space. Note that for non-overlapping windows, every window is simply a document, and since non-overlapping windows are used for evaluation (as in the original paper), we will address them as documents interchangeably.

Let f be the number of features extracted as described in table 1. The writeprint patterns creation is as follows:

1. Calculate information gain for all f features as described in equation 1.
2. For every author A do the following steps:
   Let d be the number of documents of author A.
   (a) Extract all f features from all d documents of author A to construct the feature matrix X_A:
   $$X_A = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1d} \\ x_{21} & x_{22} & ... & x_{2d} \\ ... & & & \\ x_{f1} & x_{f2} & ... & x_{fd} \end{bmatrix}$$
   (b) Maintain indices of all zero-frequency features of author A – features that were never used in any of the d documents.
   (c) Compute the covariance matrix of X_A, denoted Σ_A.
   (d) Calculate the **basis matrix** B_A out of the covariance matrix Σ_A which is the set of $n \leq f$ eigenvectors $\{u_i\}$ corresponding to eigenvalues $\{\lambda_i\}$ with values $\lambda_i > 1$. B_A's dimension is $f \times n$.
   (e) Compute the **writeprint** W_A, an $n \times d$ matrix which is the projection of A's feature matrix (i.e. set of d feature value vectors) onto an n-dimensional space – the principal components matrix:
   $$W_A = B_A^T X_A$$

The eigenvectors of the covariance matrix are taken as the basis matrix since they maximize the variance of the projections of all input vectors in the feature matrix X_A.

3.3.1 Implementation Notes

In the basis matrix construction, when restricting the selected eigenvectors to only those with eigenvalues greater than 1, the basis matrix ended up empty. Even when testing on small sample-size matrix with less than 20 features (as opposed to more than 1200 in the evaluation experiments), only 2 eigenvalues held-up to the condition. Therefore in this implementation all eigenvectors are taken, i.e. no space reduction is applied. However, although all features are taken into account and not just those with the greatest variance, the evaluation resulted with high accuracy rate (see section 4).

For covariance calculation, an unbiased estimator was used [3] to calculate the $f \times f$ matrix:

$$\hat{\Sigma}_A = \frac{1}{d-1} \sum_{i=1}^{d} (X_i - \mu_i)(X_i - \mu_i)^T$$

[3] http://en.wikipedia.org/wiki/Estimation_of_covariance_matrices

Where X_i is the i-th document of the f-dimensional random variable and $\mu_i = \frac{1}{d}\sum_{i=1}^{d} X_i$ is the sample mean.

In addition, the user has the option of choosing whether to hold a feature matrix including vectors for all the documents, or just one feature vector which is the average vector across all documents. Since this option significantly increases performance, it is set as default. In addition, when tested on the Brennan-Greenstadt Adversarial Corpus, the averaged features method yielded better results (see section 4 for more details).

The implementation uses Weka's InfoGain calculation for the first phase of this part, to be used in the next phase – calculating pattern disruption.

Lastly, the phase above is done for every training author using all of his training documents. For the anonymous test documents, the calculation differs between cross-validation and standard train-test: for cross validation, the same process is applied on the test documents, i.e. for each author whose documents are to be tested, all of the test author's documents are taken to build that author's writeprint. For train-test evaluation, each anonymous test document is taken on its own, as if it is a single document belonging to some tested author (i.e. no knowledge of similar authorship between any two test documents is incorporated).

3.4 Pattern Disruption

Since the Writeprints method is based on individual-author-level features, features an author never uses in any of his documents may be important – if an anonymous document contains features not used by the author, this document should be considered stylistically less similar to this author's style.

For any two authors A and B, since they both use different feature sets, two comparisons are required: a pattern for B constructed from B's text, A's feature set and A's basis matrix, denoted P_B^A, to be compared with W_A, and vice versa – P_A^B to be compared with W_B. We will examine the comparison of P_B^A with W_A: in order to take into account features that A never uses but B does, we add "pattern disruptors" to A's basis matrix B_A.

For every training author A and test author B, and vice-versa, pattern disruptors are added as follows:

1. For every feature p that has zero frequency in A's documents but non-zero frequency in B's documents:
   (a) Calculate the pattern disruption value δ_{kp} for every eigenvector u_k in the basis matrix B_A ($k = 1, ..., n$):

   $$\delta_{kp} = IG(c, p)K(syn_{total} + 1)(syn_{used} + 1) \tag{2}$$

   Where $IG(c, p)$ is the information gain for feature p across the set of author-classes c, syn_{total} is the total number of synonyms for the feature p and syn_{used} is the number of synonyms used by the author. The synonym-count is done only for word-based features, and for all other features $syn_{total} = syn_{used} = 0$. K is a disruptor constant to control the disruption magnitude, and is set to 2, according to the original paper.
   (b) Determine δ_{kp}'s sign for every $k = 1, ..., n$ as follows:

   $$\delta_{kp} = \begin{cases} -\delta_{kp} & \text{if } \sum_{i=1}^{d_A} \frac{W_{A_{ik}}}{d_A} > \sum_{i=1}^{d_B} \frac{P^A_{B_{ik}}}{d_B} \\ \delta_{kp} & \text{otherwise} \end{cases} \tag{3}$$

   (c) For all $k = 1, ..., n$, append δ_{kp} to eigenvector u_k in B_A.

The information gain and synonymy usage in equation 2 is incorporated in the pattern disruption calculation in order to take into account feature stability – how often does the feature change across authors and documents. This way words that are less reflective of style (and more of content) have less influence.

The sign correction in equation 3 is intended to have those values in B_A divert P_B^A farther away from W_A. For instance, if for some dimension k, W_A is located to the left of P_B^A, then δ_{kp} will be set to be positive in order to move P_B^A farther away from W_A.

3.4.1 Implementation Notes

Since the implementation derives basis matrices of dimension $f \times f$ (including all eigenvectors), pattern disruptions are not appended to the basis matrix, but set to the corresponding feature indices in the basis matrix.

The synonymy usage calculation is implemented using Wordnet [14] as used in the original paper, however the synonymy count is not clear from the paper. Since some words may appear as different parts-of-speech, a heuristic is adopted where the part-of-speech that generates the most synonyms for the word is taken, and the synonymy count is calculated with respect to it (ideally every word-based feature should have been saved with its POS tag to distinguish words that are the same but with different POS, e.g. "run"-(NN) and "run"-(VB); This is left for future work). Moreover, the user has the option of either count all synonyms in all synsets generated by Wordnet for the selected POS, or count only synonyms of the first synset.

In addition, the user has a choice of completely eliminating synonymy count calculation for word-based features, thus having syn_{count} and syn_{total} in equation 2 be equal to 0, as for any non-word-based feature. This method has shown to achieve better accuracy in this implementation, and adopted as default.

3.5 Similarity Detection

In this section the entire process is summarized including comparison identification:

1. Extract all features described in section 3.2 from all training and test documents.
2. Calculate feature matrix X_A, basis matrix B_A and writeprint matrix W_A for all authors A in the training and test set, as described in section 3.3.
3. For every training author A and test author B:
   (a) Add pattern disruptors to B_A with respect to A's zero features that are non-zero for B, as described in section 3.4. In a similar manner add pattern disruptors to B_B.
   (b) Calculate $P_B^A = B_A^T X_B$ and $P_A^B = B_B^T X_A$.
   (c) Compute the Euclidean distances $d_1 = \|W_A - P_B^A\|$ and $d_2 = \|W_B - P_A^B\|$.
   (d) Compute the average distance $d_{AB} = \frac{d_1+d_2}{2}$.
   (e) Determine B's identity as $\underset{A}{\operatorname{argmin}}(d_{AB})$ across all training authors A.

4 Evaluation

The Writeprints implementation evaluation is done in two phases.

In the first phase, the implementation is evaluated on the Brennan-Greenstadt Adversarial Corpus [2], which contains 13 authors with approximately 5,000 words per author, in documents divided into around 500 words per document. The corpus also contains adversarial documents, where the authors try to 1) obfuscate their writing style or 2) imitate the style of another author (Cormac McCarthy). The adversarial documents were omitted for the experiments conducted with this corpus. It is evaluated using cross-validation with varying configurations for optimization, to be later evaluated on a larger corpus.

In the second phase, the implementation is evaluated with optimal configuration (as resulted from the first phase) on the Extended Brennan-Greenstadt Adversarial Corpus [2]. This corpus is an extension of the previous. It includes 45 authors with at least 6,500 words per author, also divided into around 500 words per document. It also contains adversarial documents as the previous. The corpus is evaluated with both cross-validation on the non-adversarial documents and in adversarial settings: training on the non-adversarial documents and testing on the obfuscating- and imitating-attack-documents.

The results of the Writeprints evaluation in the second phase are compared with SVM classification results, using Weka's sequential minimal optimization (SMO) SVM with a polynomial kernel. Different kernels were evaluated with the corpus used for the first phase using 10-fold cross-validation,

and the polynomial kernel was selected as it yielded the highest accuracy. The feature set used are identical to those used for the Writeprints experiments (in fact, the exact same data files were used after the feature extraction phase).

4.1 Original Evaluation

The authors of the original paper evaluated Writeprints on 4 different online datasets, using 25, 50 and 100 authors: Enron emails, eBay comments, Java forum and CyberWatch chat. For comparison, they evaluated the datasets using Ensemble SVM with individual-identity-level feature sets and SVM with author-group-level feature set (they also use SVM with feature set consisting of only static features, but it was consistently outperformed by the others). The results attained for Writeprints and SVM for 50 authors are presented in table 2. It can be seen that Writeprints performs inconsistently across different datasets, however achieves accuracy significantly higher than random chance. The results for 50 authors are presented, as they are the closest in number of authors to the 45-author corpus that is used for evaluation in this paper.

Table 2: Original Writeprints and SVM accuracy (%) results for 50 authors.

Dataset	Writeprints	SVM
Enron Email	90.4	86.6
eBay comments	95.2	93.8
Java forum	66.4	86.6
CyberWatch chat	42.6	33.3

4.2 Evaluation Methods

As mentioned before, the implementation supports both cross-validation and standard train-test settings. For cross-validation, the set of documents is divided into 10 randomly generated, equally-sized subsets (folds). As opposed to standard cross-validation, where each of the folds is taken as the set of test documents with the other 9 taken as training documents, here the evaluation is conducted as in the original paper: for each experiment, half consecutive folds are taken for training and the other half for testing, e.g. fold 1 through fold 5 versus the others, fold 2 through fold 6 versus the others etc. In each experiment, for both the training and the test data, all documents of the same author are incorporated in that author's Writeprint calculation, i.e. knowledge of similar authorship of documents is taken under consideration for both. This cross-validation evaluation method is discussed in section 4.3.1.

The overall accuracy is evaluated by the averaged classification accuracy across all experiments, where:

$$\text{Classification Accuracy} = \frac{\text{Number of correctly classified authors}}{\text{Total number of authors}} \tag{4}$$

For train-test configuration, the training documents are used for calculating each author's Writeprint, where all documents of the same author are used for that author's training. However, as opposed to cross-validation, the test data is learnt by looking at each test document as if it is a single document of some anonymous author, i.e. knowledge of similar authorship between test documents is not taken under consideration. Effectively it doesn't matter in the experimental setup in this paper, as most authors have only a single obfuscation / imitation document to be tested.

4.3 Phase 1: Configuration Optimization

The Writeprints implementation presented in this paper provides some user-defined configurations that have affect on accuracy. Moreover, and perhaps even more important in real-world applications, the different configurations affect performance. The following details different configurations evaluated with the 13-author corpus using cross-validation. In each of the following experiments, one configurable attribute is tested, whereas the other attributes are set to one baseline which is the configuration that yields best performance.

4.3.1 Cross-Validation Evaluation

As mentioned, the cross-validation technique used in this section, as used in the original paper, utilizes half the folds for training and half for testing. In order to examine this method of evaluation, both Writeprints and SVM were tested on the two methods: 1) the standard 9-versus-1 folds cross-validation, and 2) the half-versus-half folds cross-validation, as in the original paper. It should be noted that it is not clear whether the original authors used the same cross-validation method for their SVM experiments. The Writeprints configurable attributes are set to the following baseline: 1) Using averaged feature vectors, 2) reducing the feature space by the top-used heuristic, and 3) *not* using synonym-count information for pattern disruption calculation of word-based features.

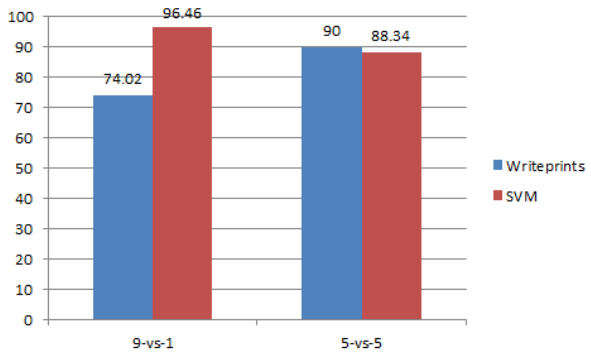

Figure 1: 13-authors 9-versus-1 and half-versus-half cross-validation accuracy results.

Accuracy results are summarized in figure 1. Using the half-versus-half cross-validation yielded 90% and 88.34% accuracy for Writeprints and SVM, respectively. Although Writeprints outperformed SVM, the results are insignificant (with $p < 0.05$). Using the standard 9-versus-1 cross-validation yielded 74.02% and 96.46% accuracy for Writeprints and SVM, respectively, i.e. SVM outperformed Writeprints (significantly with $p < 0.05$).

I believe the reason for the major decrease in accuracy for Writeprints is due to the decrease in the number of feature vectors that take part in the average for all test authors' data. As the number of vectors decreases, that author's Writeprint and basis matrix become less "smooth" with respect to that author's style.

Since we are interested in examining the best knowledge that can be extracted from the given data, for the rest of the evaluation Writeprints will be examined using half-versus-half cross-validation, whereas SVM will be examined using standard 9-versus-1 cross-validation.

4.3.2 Averaged Feature-Vector

As mentioned in section 3.3.1, the implementation provides the user the option to hold one feature vector which is the average of all feature vectors extracted from all documents of the given author, instead of the entire feature matrix (all feature vectors). The other configurable attributes are set to the following baseline: 1) Reducing the feature space by the top-used heuristic (mentioned in section 3.2.1), and 2) *not* using synonym-count information for pattern disruption calculation of word-based features.

Results: The averaged feature-vector configuration outperformed the non-averaged in terms of both accuracy and performance. In terms of accuracy, the averaged method attained 90% accuracy, outperforming the non-averaged feature matrix method, which attained 84.62% accuracy. In terms of performance, the averaged method significantly outperformed the other and took only a fraction of the time the non-averaged took.

4.3.3 Reducing Feature-Space to Top-Used

As mentioned in section 3.2.1, the user has the option of reducing the feature-space by applying information-gain on the feature-classes with potentially high number of features (e.g. letter n-grams,

bag-of-words, etc.) and removing all features except for some fixed number of features with the top information-gain values. The second option the user has is to apply the top-used heuristic, by which the feature classes mentioned are narrowed down to the features with the highest frequency across the entire training corpus. The other configurable attributes are set to the following baseline: 1) Using averaged feature vectors, and 2) *not* using synonym-count information for pattern disruption calculation of word-based features.

Results: The top-used heuristic configuration outperformed the information-gain configuration in accuracy: 90% versus 86.92%, respectively. In terms of performance, there was no significant difference in the Writeprint construction phase, but as mentioned, the feature extraction phase took much longer for the information-gain configuration.

I believe a reason for this result could be initially large size of the feature space, which might be causing inaccuracies in information-gain calculation, resulting with many values be set to 0. Thus when some feature class is reduced to the information-gain top-n, there is no distinguishing between real zero-information-gain features and those falsely assigned zero.

4.3.4 Synonymy-Count in Pattern-Disruption Calculation

As mentioned in section 3.4.1, the user has the option of using synonymy-count to be incorporated in the pattern-disruption calculation phase for all word-based features. The other option is to treat all word-based features as any other feature without incorporating any synonymy-count knowledge, by setting syn_{total} and syn_{used} in equation 2 to 0. If synonymy-count information is incorporated in pattern-disruption calculation, the user can choose between counting all synonyms of synsets generated by Wordnet for the given word, or count synonyms only in the first synset. The other configurable attributes are set to the following baseline: 1) Using averaged feature vectors, and 2) Reducing the feature space by the top-used heuristic.

Results: The no-synonymy-count configuration outperformed the synonymy-count configuration, counting all synonyms and counting synonyms only in the first synset: 90% versus 70.77% and 82.31%, respectively. In terms of performance, there was no significant difference between any of the experiments.

I believe the reason for this result is the heuristic selected for the synonymy-count method mentioned in section 3.4.1, used due to the lack of a detailed method in the original paper. Incorporating synonymy-count using this method perhaps affects pattern-disruption values too much, blurring distinguishability between two distinct identities. The refinement of counting synonyms only in the first synset, thus reducing pattern-disruption values, supports this notion.

4.4 Phase 2: Evaluation Experiments

The experiments in this section were performed using the optimal configuration resulted in the first phase: 1) Using averaged feature vectors, 2) reducing the feature space by the top-used heuristic, and 3) *not* using synonym-count information for pattern disruption calculation of word-based features.

4.4.1 45-Authors Cross-Validation

In the cross-validation experiment, the Extended Brennan-Greenstadt Adversarial Corpus was evaluated on its entire 45 authors, excluding any adversarial documents (i.e. obfuscation and imitation attacks written by the authors). Accuracy results are summarized in figure 2. Both Writeprints and SVM significantly outperform random chance, which is 2.22% for 45 distinct authors, with 74.89% and 89.84% accuracy, respectively. It can be seen that SVM outperforms Writeprints in terms of accuracy. In addition, SVM performed much faster than Writeprints.

4.4.2 Obfuscation and Imitation Attacks

In the obfuscation attack experiment, the classifiers were trained on the non-adversarial documents of the entire 45 authors in the corpus, and evaluated on the obfuscation-attack documents. In these documents, the authors attempt to hide their writing style by changing it.

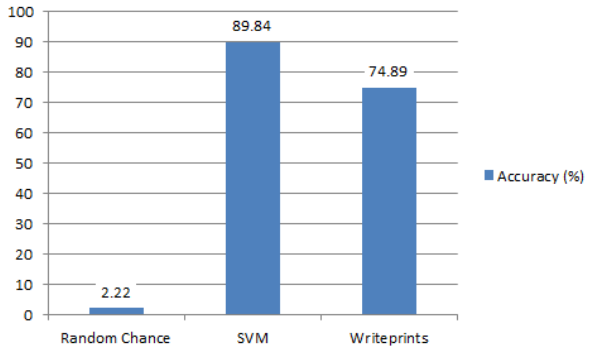

Figure 2: 45-authors cross-validation accuracy results.

In the imitation attack experiment, the classifiers were trained the same, and evaluated on the imitation-attack documents. In these documents, the authors try to imitate the writing style of the author Cormac McCarthy.

Accuracy results for both experiments are summarized in figure 3.

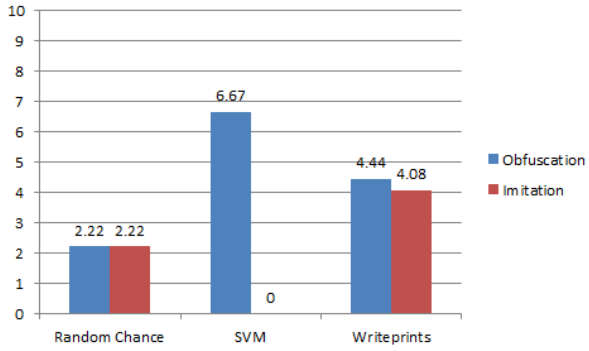

Figure 3: 45-authors obfuscation- and imitation-attack accuracy results.

The results show that both SVM and Writeprints (with the extensive feature set) are not robust against both obfuscation and imitation attacks, and behave close to random chance.

5 Conclusions

This paper presented an implementation of the Writeprints supervised authorship attribution method, as an additional analyzer of JStylo, an authorship-attribution research platform. The implementation deviates from the original method by some user-configurable options, as described in section 3. These modifications simplify the implementation, cover issues that were not clear in the original paper, and introduce options that may significantly increase performance.

The implementation is evaluated on the Extended Brennan-Greenstadt Adversarial Corpus, and shown to achieve high accuracy in non-adversarial settings, for a large number of authors (45). However, it also shown to be prune to obfuscation and imitation attacks, i.e. its performance was reduced close to random-chance in adversarial settings, where the authors successfully changed their writing style.

Another important result to be noted is the performance Writeprints achieved compared to SVM, with the same author-group-level feature set. It appears that in terms of both accuracy and performance, SVM outperforms Writeprints. It should be noted that these results may be limited to the type of writings in the corpus used for evaluation, as the original Writeprints evaluation showed

across different datasets (in section 4.1). In addition, the use of individual-identity-level feature sets should also be examined for possible accuracy improvement.

Suggestions for future work:

- Evaluate the implementation on other types of writings (e.g. all sorts of online communication, like emails or forum messages).
- Develop other heuristics for synonymy usage calculation, to be beneficial for increasing accuracy, as opposed to the heuristics presented in this paper (and the current implementation).
- Examine methods for dimension reduction.
- Introduce implementation improvements, like concurrency, in order to increase performance.
- Support individual-identity-level feature sets.

References

[1] Abbasi, A., Chen, H.: Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. ACM Trans. Inf. Syst. **26**(2) (2008) 1–29

[2] Brennan, M.R., Greenstadt, R.: Practical attacks against authorship recognition techniques. (2009)

[3] Oakes, M.P.: Ant colony optimisation for stylometry: The federalist papers. In: Proceedings of the 5th International Conference on Recent Advances in Soft Computing. (2004) 86–91

[4] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter **11**(1) (2009) 10–18

[5] Juola, P.: Authorship attribution. Found. Trends Inf. Retr. **1**(3) (December 2006) 233–334

[6] Matthews, R.A.J., Merriam, T.V.N.: Neural computation in stylometry i: An application to the works of shakespeare and fletcher. Literary and Linguistic Computing **8**(4) (1993) 203 –209

[7] Merriam, T.V.N., Matthews, R.A.J.: Neural computation in stylometry ii: An application to the works of shakespeare and marlowe. Literary and Linguistic Computing **9**(1) (1994) 1–6

[8] Tweedie, F.J., Singh, S., Holmes, D.I.: Neural network applications in stylometry: The *federalist papers*. Computers and the Humanities **30**(1) (1996) 1–10

[9] Abbasi, A., Chen, H.: Applying authorship analysis to extremist-group web forum messages. IEEE Intelligent Systems **20**(5) (2005) 67–75

[10] Koppel, M., Schler, J.: Ad-hoc authorship attribution competition approach outline. Ad-hoc Authorship Attribution Contest, (P. Juola, ed.), ACH/ALLC (2004)

[11] Zheng, R., Li, J., Chen, H., Huang, Z.: A framework for authorship identification of online messages: Writing-style features and classification techniques. J. Am. Soc. Inf. Sci. Technol. **57**(3) (February 2006) 378–393

[12] Clark, J.H., Hannon, C.J.: A classifier system for author recognition using synonym-based features. In: Proceedings of the artificial intelligence 6th Mexican international conference on Advances in artificial intelligence. MICAI'07, Berlin, Heidelberg, Springer-Verlag (2007) 839–849

[13] Quinlan, J.: Induction of decision trees. Machine learning **1**(1) (1986) 81–106

[14] Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM **38** (1995) 39–41