What You Submit is Who You Are: A Multi-Modal Approach for Deanonymizing Scientific Publications

Mathias Payer, Ling Huang, Neil Zhenqiang Gong, *Student Member, IEEE*, Kevin Borgolte, Mario Frank

***Abstract*—The peer-review system of most academic conferences relies on the anonymity of both the authors and the reviewers of submissions. In particular with respect to the authors, the anonymity requirement is heavily disputed and pros and cons are discussed exclusively on a qualitative level.**

In this paper, we contribute a quantitative argument to this discussion by showing that it is possible for a machine to reveal the identity of authors of scientific publications with high accuracy. We attack the anonymity of authors using statistical analysis of multiple heterogeneous aspects of a paper, such as its citations, its writing style, and its content. We apply several multi-label, multi-class machine learning methods to model the patterns exhibited in each feature category for individual authors and combine them to a single ensemble classifier to deanonymize authors with high accuracy. To the best of our knowledge, this is the first approach that exploits multiple categories of discriminative features and uses multiple, partially complementing classifiers in a single, focused attack on the anonymity of the authors of an academic publication.

We evaluate our author identification framework, deAnon, based on a real-world data set of 3,894 papers. From these papers, we target 1,405 productive authors that each have at least 3 publications in our data set. Our approach returns a ranking of probable authors for anonymous papers, an ordering for guessing the authors of a paper. In our experiments, following this ranking, the first guess corresponds to one of the authors of a paper in 39.7% of the cases, and at least one of the authors is among the top 10 guesses in 65.6% of all cases. Thus, deAnon significantly outperforms current state-of-the-art techniques for automatic deanonymization.

I. Introduction

In academia, the publication process and the merit of a publication is often based on rigorous peer review. In computer science, many conferences rely on a double-blind review process, where both the authors of a submission and the reviewers of a paper stay anonymous. The motivation for hiding the author's identity during the review process is to reduce any implicit or explicit bias a reviewer might have against an author or an author's affiliation. The motivation to hide the reviewer's identity is to provide proper feedback, to support him or her in asking crucial but potentially unpleasant questions, and for him or her to be protected from any attacks by authors who feel disadvantaged or wrongfully rejected.

In order to ensure anonymity, the authors of a paper are required to reformat their paper prior to submission by (i) removing the names of authors and their affiliations from the title section, (ii) describing all previous work in the third person, and (iii) blinding particular references if anonymity is not guaranteed otherwise. Generally, this process is very cumbersome for the authors and often done after the paper is already near completion. Sometimes, authors forget to conceal some revealing references or forget to rephrase parts of the paper that reveal information on some of their identities. But even if authors remove their names, describe their work in the third person, and blind their references properly, it is a common belief that a knowledgeable reviewer can correctly identify some of the authors of a paper with high accuracy. Realistically, the reviewers work in related fields and therefore they are likely to be aware of the current focus and current projects of their peers (e.g., through discussions at conferences, grant proposals, or resubmissions of rejected papers). However, this belief requires experimental validation.

Several prior studies [5], [15] showed *some* success on author identification for scientific papers using *only* the citations made in the paper. However, their success was mostly achieved in constrained space, e.g., identifying authors for papers in specific domains (e.g., Physics [15] or Machine Learning [5]), or for papers sharing common (self-)citations [5]. Citations are just one source of information and, to make a strong quantitative argument about author anonymity, one should take into account all the information that is available to reviewers.

We overcome these limitations by incorporating additional heterogeneous information that is available in scientific papers. This turns out to be a challenging task. Although additional features derived from writing style and contents of the paper are available in anonymous submissions, it is difficult to combine them with citation features to improve accuracy significantly over citation-based author identification. Naturally, features derived from different types of information differ from each other, show different sparsity, and are at different scale. We show in our evaluation that simply concatenating all features regresses the overall results and reduces accuracy.

To make things worse, a scientific paper often has multiple authors, each of them adding their own footprint to the paper, which makes the problem much more difficult to model as a whole. While the range of topics in an academic setting is generally narrow, different authors may write papers on the same topic, and the same author may change research topics over time, which, in turn, makes it a challenging task to leverage content information to identify authors.

To address these challenges, we introduce *deAnon*, a framework for breaking the anonymity of authors of anonymized academic papers. We target authors that published at least 3

Manuscript created Oct. 31, 2014; received Mar. 17, 2014; revised Jun. 20 2014; accepted Oct. 29, 2014. This work was supported by the NSF under Grant CCF-0424422 and by the AFOSR through the Multidisciplinary University Research Initiative (MURI) under Award FA9550-09-1-0539.

Mathias Payer is with Purdue University, Ling Huang is with Datavisor Inc, Neil Gong is with UC Berkeley, Kevin Borgolte is with UC Santa Barbara, Mario Frank is with UC Berkeley.

papers in the past, and these papers are then used to train multi-label classifiers to guide the attack. Our approach then attacks the anonymity of the established authors.

The *deAnon* framework uses a large historical corpus of papers in the Portable Document Format (PDF), the most common distribution format for academic publications, as the input to extract any relevant information, including but not limited to title, authors, text, citations, and other information. This textual data is then leveraged to extract specific features based on writing style, topic, and citations for each author. We adapt several multi-label, multi-class machine learning methods to model the patterns exhibited in each feature category, and propose a principled way to combine them into an ensemble classifier to aggregate complementary information. This final classifier is then used to break the anonymity of the respective author for new, anonymized submissions.

The main contributions of our paper are the following:

1) We provide a *systematic, large-scale study* on breaking the authors' anonymity of submissions based on 3,894 papers from 17 different computer science conferences[1]. Our results show that a machine can reveal an author of a paper on the first guess with 39.7% accuracy and at least one author is among the first 10 guesses with 65.6% accuracy. Our results confirm the anecdotal belief that it is possible to guess authors with high accuracy given prior knowledge of publications of these authors.
2) We design and implement *deAnon*, the first *multi-modal* attack framework that learns per-author ensemble classifiers based on writing style, topics, and citations. Furthermore, we solve the problem of combining these heterogeneous sources of information about an author in one unified framework.
3) We discuss high-profile features and common pitfalls that result in an immediate loss of anonymity for authors in a scientific context (e.g., tools that embed user names and author names in PDF files). We present possible remedies that improve anonymity by leveraging our attack framework in a feedback-guided process.

The remainder of the paper is organized as follows: first, in Section II, we discuss the design of our framework, then, in Section III, we discuss data and feature extraction. Following, in Section IV, we provide details on the prediction engine and, in Section V, we evaluate *deAnon* based on a real-world data set of academic papers from various top-tier conferences. Future work based on *deAnon* and pitfalls one has to take care of when authoring a paper are discussed in Section VII. In Section VIII, we compare *deAnon* to related work, and, finally, we conclude in Section IX.

II. deAnon Design

Our approach aims to simulate a realistic peer-review setting. We assume that an attacker knows and possesses or has access to a large corpus of published papers in the related fields and knows the names of the authors (e.g., by crawling the conference websites or digital libraries and downloading large sets of publications). The attacker extracts all kinds of information from the paper corpus to build models guiding the attack. Given an anonymous paper, the attacker extracts information from the paper, feeds it to the model, and gets the most likely list of candidate authors (ranked by some criterion). Using the ranked candidate list, he/she can then identify the authors of the paper (possibly by incorporating other information that is not part of the model). In our setting, the attacker is only interested in deanonymizing already known authors, i.e., those who have already published a number of papers. Accordingly, we define the paper deanonymization problem as follows:

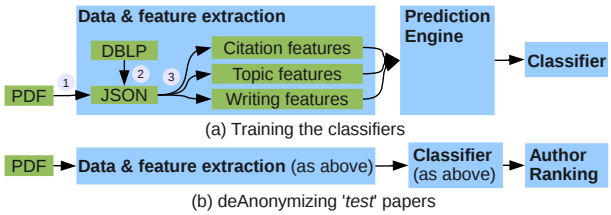


Fig. 1: Data processing pipeline for (a) training papers and (b) test papers: (1) parsing papers, (2) cleaning title section and references, and (3) extracting features.

Paper Deanonymization Problem. *Given a large corpus of historical papers and their authors and given an anonymous paper, correctly identify one or more authors of the paper from the set of all authors in the corpus.*

A. Approach Overview

Figure 1 illustrates the complete deAnon pipeline. At a high level, deAnon deanonymizes submitted papers by comparison with older papers using machine learning techniques. The input to the deAnon framework is a data set of publicly available papers published in the past years at different conferences (in the PDF format). The query data is a set of anonymized papers. Our parser recovers structured text from the PDF automatically, which is then grouped into title, a list of authors, abstract, introduction, related work, conclusion, a list of citations, and remaining sections. Our framework then transforms the structured data into vectors in the feature space and trains classifiers that capture the publishing patterns of different authors.

On completion of training the classifier, the framework can deanonymize anonymized papers in the PDF format. Given an anonymous paper, the deAnon classifier outputs a prediction score for each candidate author that indicates how likely this candidate is (one of) the true author(s) of the paper. These prediction scores are then used to rank the candidates to make a sequence of educated guesses, from most likely to least likely true authors of the paper. When making an ultimate guess about authorship, the attacker may incorporate additional characteristics or information that is not available to our automatic method. If an author appears near the top of the resulting list of ranked candidates, the author is considered especially identifiable.

[1]Anonymized submissions are not openly available. To remedy this problem we split the data corpus into a train and test data set (similar to related work), removing the names and affiliations of papers in test data.

B. Input Data

A large corpus of training data is crucial for deAnon to succeed. For each identifiable author we need papers that were written by her or him. deAnon cannot identify authors without prior publications in our training data set because it cannot learn models for these authors.

In academia, the camera-ready version of most papers is published and publicly available from the author's or institute's website. In addition, most venues release published papers in digital form. We use this data to train the classifiers. Experienced authors, e.g., professors, have published multiple papers as part of their own doctoral studies, during their tenure process, or when advising their students. We observe that a large portion of papers are authored or co-authored by at least one experienced author. Generally, the data set should be large enough to have prior publications for the experienced authors of the fields.

III. Data and Feature extraction

In this section, we describe the design choices made to extract data from scientific papers and present in detail how we derive three categories of features from the extracted data.

A. Data Extraction

Some conferences and authors release papers in a structured format, e.g., HTML. Unfortunately, there is no unified structured format for which more than a couple of papers are available and it is practically infeasible to implement a converter for each format and conference. For example, USENIX Security published many papers in structured file formats, but the format varies from year to year.

In contrast, papers are almost always available in PDF format. However, PDFs are hard to analyze because the format focuses on the layout of the paper's pages rather than its text flow[2]. Our parser recovers the missing structural information using layout analysis and identifies the heading section, which includes title and list of authors (for training papers), the citations, and individual text sections like abstract, introduction, related work, conclusions, and remaining text.

As it turns out, it is very hard to extract citations from a corpus of PDF files. The main reason is that in order to be useful for author prediction, a citation must be matched with an author in the database of all cited authors of the data set and with papers of the database, if it exists there. This matching step is hindered by different citation formats, abbreviated names, different ordering of first name and last name. Therefore, our parser matches the authors, title, and parsed citations of each publication against the publicly available DBLP [20] data to recover a clean version of the meta-information.

B. Feature Extraction

Table I gives an overview of the different features used in our classification models. Using structured data, we extract features that can capture the characteristics of the individual authors to train machine learning classifiers. deAnon uses features from three different categories of information available in the structured data: the *writing style* of the paper, the *topic* of the paper, and the *citations* made in the paper.

Category	Description	Count
Writing style		
Length	number of words/characters	2
Vocabulary richness	Yule's K and frequency of *hapax legomena*, *dis legomena*, etc.	11
Word shape	frequency of words with a mix of upper and lower case letters	5
Word length	frequency of words that have 1–20 characters	20
Letters	frequency of *a* to *z*, ignoring case	26
Digits	frequency of 0 to 9	10
Punctuation	frequency of `.?!,;:()"-'`	11
Special characters	frequency of other special characters `` `~@#$%^&*_+=[]{}|/<> ``	21
Function words	frequency of words like 'the', 'of', and 'then'	293
Content		
Bag of words	concatenating text from paper title, abstract, and conclusion sections, removing stop words, doing word stemming and selecting word stems that have a frequency of at least 20 as features.	2,374
Citation		
References	each paper is treated as a binary feature.	7,954

TABLE I: Features used for classification. In total, we have 10,727 heterogeneous features.

1) Writing Style Features: The writing style of an author describes the manner in which an author encodes information for the audience. Identifying the author of a text based on the writing style has an established history [24], [25], [1], [27].

We extract writing style features for each paper from its text, excluding its references. Following the work by Narayanan et al. [27], the extracted writing style features reflect the distributions of characters and words such as the frequency of the letters and the frequency of function words.

2) Topic Features: Unlike blog authors who may cover substantially different topics from post to post, scientific researchers generally focus on a core research area in which they publish a series of papers. Thus, it is reasonable to assume that the central topics of a paper correlate with the core areas of a researcher. Therefore, we assume that the topic of a paper is informative for identifying its authors.

There are several ways to capture topic features of the papers. A straightforward way is topic modeling on the corpus of plaintext extracted from the paper [4], [30]. However, due to the rich structure of research papers, some sections of a paper, i.e., title, abstract, and conclusion, may have more information to indicate the topic of a paper than the other sections (e.g., related work or evaluation). For any well-written paper, we believe that its text in title, abstract, and conclusion sections was carefully drafted by the authors to capture the main ideas and contributions of the paper in the best possible way, thus the text in those sections approximates the main topic of the paper well. Therefore, instead of performing topic modeling,

[2]The PDF format does not define a document structure but only focuses on the placement of characters. PDF supports embedded images and some publications are only available as scanned versions, relying on an OCR step.

we leverage a bag-of-words model to extract textual features from the title, abstract, and conclusion sections.

Moreover, we build a corpus out of text from title, abstract, and conclusion sections from all papers, and finally, leverage the MATLAB Natural Language Processing tool (MatlabNLP [14]) to extract the bag-of-words features. During extraction, we perform stop-word removal and word stemming. Following the extraction, we normalize the features by applying the standard Term Frequency-Inverse Document Frequency (TF-IDF [28]) method.

Although we use topic features, we do not expect them to provide reliable and accurate predictions alone, particularly due to the following reasons: (i) different authors write papers on the same topic and compete with each other, i.e., two papers on a similar topic can be from distinct sets of authors; (ii) authors might change their topic over time, and write papers on topics that have little correlation to each other; (iii) highly-productive authors, such as professors advising large groups, publish papers on a diverse range of topics, and even in different areas. Therefore, we imagine topic features to be complementary to other more fundamental features. Combining them together however, we can achieve better results than by leveraging only individual features.

3) Citation Network Features: Scientific publications use citations to refer a reader to related or similar previous work, generally because the work to be presented builds upon this earlier work or is compared to this work. A popular belief is that the list of references provides hints on the authors of a paper. Two main assumptions support this belief. First, it is assumed that authors tend to cite their own papers. One reason for this might be that authors have a better overview over their own prior work than over the entire literature. Clearly, another reason might be that citing a paper increases the paper's impact which is in the interest of the author. Second, it is often assumed that authors use similar citations for many of their papers. Since a thorough literature review for a given topic is time intensive, authors might carry out this review only for their first paper on a new topic and then later re-use their findings for other papers on this topic. Consequently, a large fraction of citations may be shared across papers of the same author and the same set of papers that have not been cited in earlier papers might have been missed in newer papers as well.

An important consideration for the citation network feature is the informational content of each citation. If a paper is cited by many different authors in different papers, i.e., if it is a high-impact paper, then the informative value of this citation is not as high as if the paper is of low-impact. Popular and high-impact papers are well known by the academic community and cited often. Rare papers, on the other hand, are only known by few people and, in turn, leak more information about the specific authors or their affiliations.

In earlier work [5], [15], citations have been used *as the only feature* to deanonymize authors of scientific papers and have been found to be highly discriminative, confirming prior beliefs of researchers who participate in the peer-review process. Therefore, we include the citations of a paper as *one of the* features in our classification framework.

IV. Prediction Engine

In this section we describe our prediction engine that takes the extracted features of a submitted paper as an input and outputs the prediction for the authors of this paper. We begin by explaining the task from a general machine learning point of view, and follow-up by describing the various parts that our approach leverages. Some of these individual classifiers are off-the-shelf methods; some are tailored to the author prediction problem. One of the biggest challenges that we face is to combine these multiple classifiers to achieve a significantly better prediction accuracy than any individual one and prior work.

A. General Machine Learning Setting

We model the author prediction problem as a **multi-label classification problem**, i.e., a paper can have more than one labels (authors). Suppose we have m authors, and we denote them as $A = \{a_1, a_2, \cdots, a_m\}$. Moreover, we denote a paper, its feature vector, and its labels as p_i, $\mathbf{p}_i \in \chi$, and $\mathbf{a}_i \in \{0,1\}^m$, respectively, where χ is the feature space and $a_{ij} = 1$ ($j \in \{1, 2, \cdots, m\}$) means that the author a_j coauthored p_i. In the training phase, a set of papers whose features are $\mathbf{p}_i^{(\text{tr})}$ and labels are $\mathbf{a}_i^{(\text{tr})}$ are given to train a model, where $i = 1, 2, \cdots, n^{(\text{tr})}$. In the test phase, we are given a set of unlabeled testing papers whose features are $\mathbf{p}_i^{(\text{te})} \in \chi$, where $i = 1, 2, \cdots, n^{(\text{te})}$. In contrast to conventional settings where an estimated binary author vector is outputted, our model produces a score vector $\mathbf{s}_i$ for each test paper $p_i^{(\text{te})}$, where a score entry $\mathbf{s}_{ij}$ corresponds to the likelihood that $p_i^{(\text{te})}$ is co-authored by a_j. After predicting all $n^{(\text{te})}$ test papers, we combine all prediction scores to obtain a score matrix $S \in \mathbb{R}^{n^{(\text{te})} \times m}$.

Figure 2 illustrates the prediction engine in detail. In the following section, we describe the different classifiers that operate on the introduced features and handle this multi-label problem. Our prediction engine then combines the output of these individual predictors into a single prediction using ensemble learning.

B. Citation Classifiers

In Section III-B3, we argue that the references made in a paper are a valuable source for predicting the authors. In this section, we investigate different methods to leverage this information on citations.

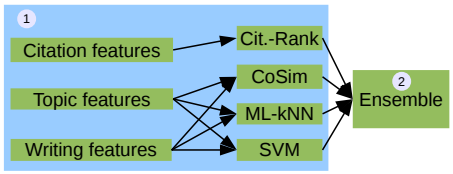


Fig. 2: Workflow of the prediction engine: (1) classifier training, (2) combining the classifiers using the ensemble method.

1) Top-cited Author: The simplest guessing strategy is to select the authors that are cited the most often in the paper. While this method has been proven successful in previous work [5], [15], it bears the drawback that it outputs only a small set of candidate authors that are randomly ordered and it provides no estimate on other potential authors. In our experiments, we refer to this predictor as *top-cited author*.

2) Author Citation Rank: To overcome the limitations of the *top-cited author* predictor, we introduce a predictor that augments each candidate author with a citation score. This citation score is the number of citations of the author's work, i.e., the count on how often the respective author is cited in the paper. In this way, the predictor provides a list for the second and third most likely author and so on [15]. Yet, a drawback remains: authors who are not cited at all will not be assigned a score by *author citation rank*.

3) Weighted Cosine Similarity Ranking: In order to provide a prediction score for candidate authors that are not cited at all in the paper, we adopt a method proposed by Bradley et al. [5]. Their method establishes a neighborhood relation between all papers in the training set and all papers in the test set. With m candidate authors, the feature vector of a paper p_i is $\mathbf{p}_i \in \mathbb{N}^m$, which indicates for each author how often he or she has been cited in p_i. For two papers whose features are $\mathbf{p}_i, \mathbf{p}_{i'} \in \mathbb{N}^m$, the method computes the cosine similarity to measure how close these papers are: $S(\mathbf{p}_i, \mathbf{p}_{i'}) = \mathbf{p}_i \cdot \mathbf{p}_{i'} \left(\|\mathbf{p}_i\| \|\mathbf{p}_{i'}\|\right)^{-1}$.

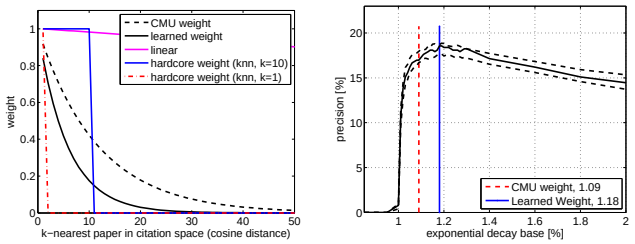
For each test paper $p_i^{(\text{te})}$, the $n^{(\text{tr})}$ scores $S(\mathbf{p}_i^{(\text{te})}, \mathbf{p}_{i'}^{(\text{tr})})$, $i' \in \{1, 2, \cdots, n^{(\text{tr})}\}$ provide a neighborhood ranking of all training papers. We define the ranking index as $t \in \{1, 2, \cdots, n^{(\text{tr})}\}$. Authors of papers that are ranked high (i.e., those that are close) receive a higher score than those with a low ranking. There are several ways to aggregate scores for each candidate author $a_j \in A$ over all the training papers. Bradley et al. [5] suggest to aggregate the ranking scores using exponentially decaying weights. We generalize this weighting scheme idea and explore several other weight functions. The general aggregation formula is:

$$\mathbf{s}_{ij} = \sum_{t=1}^{n^{(\text{tr})}} \delta(\mathbf{a}_{tj}^{(\text{tr})} = 1) \cdot w(t) \ , \qquad (1)$$

where the switch function $\delta(x)$ returns 1 if the predicate x is true and 0 otherwise. The method of Bradley et al. [5][3] is modeled with a decay base of 9% per rank position, i.e., $w(t) = 1.09^{-t}$.

We investigate the following weight functions. We explore *linearly decaying weights* with $w(t) = n^{(\text{tr})} - t$, as well as *hardcore weights* with $w(t) = 1 - \delta(t > \tau)$. For hardcore weights and $\tau = 1$, this method boils down to assigning all weight to the authors of the nearest paper in citation space. More generally, for $\tau = k$, the method assigns equal weight to all authors of the k nearest neighbors. Finally, we improve the exponential weighting scheme $w(t) = d^{-t}$ by selecting the optimal decay base d through cross-validation on the training data. We select d^*, such that it maximizes the success rate on randomly hold-out training papers and then use $w(t) = (d^*)^{-t}$ on the entire training set. We refer to this method as *learned weights*.

[3] Equation 1 was multiplied by 500,000 for Bradley et al. [5]. We omit this constant factor because it does not affect the order of guessing authors.



(a) Weighting schemes

(b) Success rate of first guess for different weights

Fig. 3: a) Weighting schemes b) Cross-validation search for optimal weighting scheme on training data.

All weighting schemes are graphically illustrated in Figure 3 (a). The x-axis indicates the paper ranking according to cosine similarity with the respective test paper. The y-axis indicates the score that will be assigned to all authors of the training paper at the respective rank. Figure 3 (b) depicts the search result for cross-validation of the exponential decay base d. We see that weighting those authors higher who have more distant papers in citation space (that is $d < 1$) leads to a 0% success chance. For $d > 1$, authors of closer papers are assigned higher weights, which immediately improves the results. The accuracy continues to increase until, for $d > 1.18$, it slowly and monotonically decreases, suggesting that it pays off to not completely ignore authors of papers that are more distant than the nearest neighbor.

C. Generic Classifiers

In this section we explain three generic classifiers, SVM, ML-kNN, and ranking by cosine similarity, that are feature-agnostic, i.e., they work independently from the particular features extracted from the data. We choose to use them because they are capable to handle high dimensional sparse data, and are widely used in multi-label classification problems [22].

1) Support Vector Machines: Support Vector Machines (SVM) are popular binary classifiers [10] and often achieve the best prediction results for tasks with high dimensional sparse data [19], [31]. It has also been shown [22] that SVM-based variants perform among the best multi-label classifiers in text-mining related settings.

We use linear SVM, which has been shown to be efficient for training large-scale problems and to exhibit similar prediction accuracy on high-dimensional sparse data as the more complex and time-consuming kernel SVM [19], [13]. We use the known one-vs-all approach to transfer our multi-label classification problem into multiple binary SVM classification problems [32]. For each candidate author, the one-vs-all approach trains a classifier that distinguishes the author's papers from all other papers. The scores of the set of all such binary classifiers are then combined to provide a score vector for the multi-class problem. In the following, we describe how each of these steps works in detail. We use the LIBSVM implementation [9] to train the involved support-vector machines.

Using the training data set, where each paper may have multiple authors, we create m classifiers, one for each author $a_j \in A$. For the classifier of a_j, we label the papers with binary labels $\{+1, -1\}$. In this data set, the positive class includes all the training papers that a_j has co-authored, and the negative class includes all the other papers in the training set. We then train a SVM classifier c_j on the training examples and predict on the testing examples. When predicting whether a test paper p_i has a_j as an author, instead of just outputting a binary prediction, we output the SVM score $\mathbf{s}_{ij}$. This score can be interpreted as the probability that the classifier of a_j predicts that paper p_i belongs to class $+1$. In total, we obtain m scores from the m classifiers for each test paper, each of which indicates how likely the given paper is authored by the respective candidate author.

2) ML-kNN: ML-kNN is a classification method that has a proven track record regarding classification performance on multi-label problems [22], [33]. ML-kNN was first proposed by Zhang et al. [34], where it was tested on three naïve classification problems.

The ML-kNN classifier works as follows: For each test paper p_i, it requires the k most similar papers $N(i)$ in the training set as an input. For all three kinds of features, we use cosine similarity as the similarity metric and compute $N(i)$ for each test paper. The classifier then computes a membership count vector $\mathbf{m}_i \in \mathbb{N}^m$, where $\mathbf{m}_{ij} = \sum_{i' \in N(i)} \delta(\mathbf{a}^{(\text{tr})}_{i'j} = 1)$. Intuitively, the element $\mathbf{m}_{ij}$ of this vector indicates how many papers in $N(i)$ are co-authored by a_j. Treating these counts as a feature, we compute the scores:

$$\mathbf{s}_{ij} \quad = \quad P\left(\mathbf{a}_{ij}=1 \middle| \mathbf{m}_{ij}\right) \tag{2}$$

3) Ranking by Average Cosine Similarity: Ranking by average cosine similarity (CosSim) slightly differs from the previous weighted ranking methods based on citations. It is also based on cosine similarity but can be applied generically to the other features and is not specific to citations. This method has been shown to perform surprisingly well even when only a small amount of training data is available [27].

Suppose we have m authors in the training data set. For each test paper, we construct a characteristic vector whose length is m, each entry of which corresponds to an author. The characteristics vector is constructed as follows: First, for each candidate author a_j, we find all the training papers that are co-authored by a_j. Second, we compute the cosine similarity in a generic feature space for all pairs of the test paper and such training papers. This feature space could either be spanned by style features, by topic features, or by citations. Third, the score of author a_j is the average of these cosine similarities. The higher this score is, the closer in feature space are the papers of this author to the target paper.

D. Combining Multiple Modalities

Given a set of heterogeneous features from multiple sources, including writing style, topics, and citations of the paper, there are a variety of ways to combine them together to learn classifiers and make predictions. A naive way is to simply concatenate all the different feature sets together into a single feature set (e.g., concatenating all rows of different feature sets together). However, we do not expect this approach to achieve the best results, due to the heterogeneous nature, sparsity, and scale of different features. This is also evident by our experimental results in Section V.

Instead, we use **ensemble methods** to gracefully combine multiple feature sets. Ensemble methods are machine learning techniques that strategically combine multiple (diverse) models to solve a particular computational intelligence problem [29]. Ensemble learning is powerful in that it can fuse together heterogeneous data from multiple sources, and/or use multiple models to obtain better performance than any of the constituent models could, especially when there is a significant diversity among data and models [7].

In our setting, we design three classifiers: CosSim, ML-kNN, and SVM for each of the three feature sets: writing style features, topic features, and citation features plus the best citation-specific classifier. As a result, we have 10 classifiers that are naturally diverse (due to both the diversity among features and models). We use a stacking ensemble approach to combine the individual classifiers together [12]. In this procedure, given c base classifiers $f_i(x)'s$ (e.g., CosSim on citation features, SVM on topic features, etc.), we construct a combined classifier that is a weighted combination of the base classifiers $f_i(x)'s$:

$$f_M(x) = \sum_{i=1}^{c} w_i \, f_i(x) \tag{3}$$

where $w_i > 0$ is the weight of hypothesis f_i, and $w_i > w_j$ means f_i has more impact to the final prediction than f_j.

To train this two-level ensemble classifier, we further divide the training part of the dataset into two equal parts. The first part is used to train individual base classifiers as described above, and the second part is used to learn the weights w_i's for combining these base classifiers. We select the w_i's that achieve the best prediction accuracy on the second part of the training data. We use a derivative-free pattern search algorithm [16] to find the optimal w_i's on the training data. This algorithm requires no knowledge of the objective function gradient. It samples points in the domain, and uses the information it has obtained to decide where to search next. This method has been applied to various cases for solving large industrial problems [8], [35].

V. Evaluation

In this section, we present and discuss our experimental findings. We first explain the methodology used to investigate our framework, describe the data used to evaluate our approach, and then analyze the results under different experimental tasks.

A. Methodology and Evaluation Metric

We simulate a realistic peer-review setting to analyze the success rate of an adversary in guessing the authors of an anonymous paper. In this setting, we use a large corpus of papers from conference proceedings and a snapshot of the

DBLP data, but no other (e.g., online) information, to build authorship prediction models.

For an anonymous paper, a prediction model outputs a prediction score for each candidate indicating how likely this candidate is (one of) the true author(s) of the paper. Using this score one can rank all the candidates for a given paper. Based on this ranking, the adversary can then narrow down the list of possible candidates to assist in manual guessing. In practice, a domain expert might achieve a high success rate by discarding some of the higher ranked candidates due to auxiliary side information. However, it is impossible to simulate this and we only simulate an adversary that blindly follows the ranking derived from our prediction model to make a sequence of guesses, where the candidate author with the highest score constitutes the first guess and the candidate with the second highest score provides the second guess and so on. We compute the probabilities that the attacker hits one of the true authors for a given number of guesses, i.e., the probabilities that the correct author is among the first K predictions.

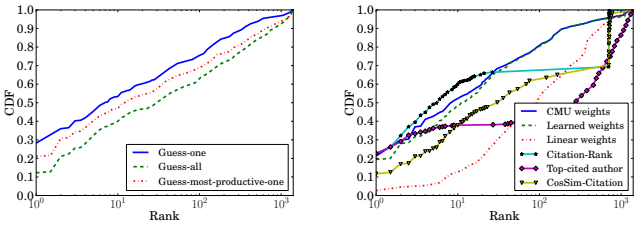
B. Dataset

Our parser extracts data from PDF versions of 8,739 papers from 17 conferences in different domains as an input data corpus. Table II lists the conferences and year that are used in this analysis. We select the span of years where digital (non-scanned) versions of the conference proceedings are available. Out of 8,739 papers, our PDF parser recovers some of the text and citations of 6,873 papers (79%). In the parsing process we drop unparsable papers (papers are dropped due to encoding problems, missing PDF features in our parser, or problems matching the format of the paper to our extraction format). For completeness, any unparsable papers may be added manually.

For each candidate author we require at least three prior publications to extract precise features. We drop papers where no author has at least three publications in our data set, reducing the data set from 6,873 papers to 5,071 papers.

Conference	years	papers	parsed	A3+	training	test
ASIACCS	06-12	224	209	120	94	15
ASPLOS	02-12	247	201	146	119	18
Usenix ATC	03-12	340	291	114	87	2
CC	02-12	322	191	113	76	3
CCS	00-12	637	578	457	357	60
CGO	03-12	317	214	155	103	8
ICNP	00-12	662	328	208	50	0
IEEE S&P	96-12	582	330	214	139	2
NDSS	98-12	300	236	151	118	4
NSDI	04-12	255	233	178	135	0
NIPS	03-12	2889	2396	2042	1291	195
OSDI	99-12	247	180	105	95	1
PLDI	02-12	420	388	307	247	30
POPL	02-12	386	333	223	148	16
SIGCOMM	02-12	397	368	282	240	20
SOSP	01-11	136	117	99	99	0
Usenix Security	98, 00-12	378	280	166	118	4
Total		8739	6873	5071	3516	378

TABLE II: Sources for our dataset. The rightmost column indicates the number of test papers with authors being undisclosed to our framework. A3+ is the number of papers where at least one author has at least 3 papers in the dataset.



(a) Per-task prediction accuracy (b) Citation classifiers

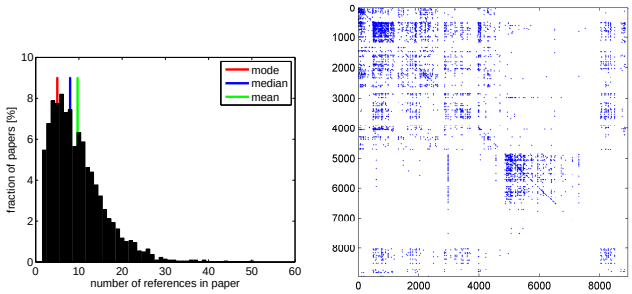
Fig. 4: a) Prediction accuracy of different tasks using a SVM classifier with writing style features. b) Accuracy of different citation classifiers.

For the feature extraction we also drop papers where some part of the paper was not parsed correctly (e.g., a missing abstract). This very conservative approach reduces our data set (training and testing) to 3,894 papers (77% of papers where at least one author has at least three papers in the dataset). There are 1,405 authors associated with these papers. These authors constitute the set of candidate authors from which we must predict when given an anonymous paper. On average each author only has less than three training papers, making it a very challenging machine learning task. Naturally, better PDF parsing capabilities and feature extraction tools will lead to more complete datasets and we expect that our prediction rate increases alongside. Similar to [27], we use three types of normalization for feature data when we train our prediction models. We find that classifiers perform differently with different normalizations. We only discuss the results with the best normalization technique due to limited space.

Unfortunately, anonymized papers are not openly available (and if they were available we would need additional ground truth about the authors and affiliations to evaluate our framework). We split unblinded data into training and testing set, removing authors and affiliations from the submission for the test data (comparable to related work). Our framework relies on features that are present both in anonymized and accepted versions of the paper. Our features are stable and depend on the writing style, topic, and citations of a submission which all do not change upon acceptance of a paper and our approach remains valid while this assumption holds. 3,516 papers from preceding years, starting in 1996, are used to train our classifiers; and 378 papers from 2012 are used to test the accuracy of our classifiers.

C. Success Rates Under Different Scenarios

We quantify the success rates of our framework under three different tasks that are defined as follows. The first task is *guess-one*, the task to correctly guess at least one of the true authors of a paper. The second one is *guess-all*, the task to correctly predict all true authors of a paper. The last one is *guess-most-productive-one*, the task to correctly guess the author of a given paper that has the largest number of publications. Intuitively, the *guess-one* and the *guess-all* task span the entire range of difficulty for the attacker. The hardest task is to guess all authors of the paper correctly, the easiest



(a) Citation statistics (b) Citation network of authors

Fig. 5: (a): Statistics of the number of references of a paper. (b): Network of citations between authors. Authors are sorted by conferences and then by year.

task is to guess at least one author correctly. In practice, an attacker would probably be satisfied with solving the *guess-one* task. Successfully guessing the group of researchers that has submitted the paper is sufficient in most cases. The real target of an attacker is the name of the group that contributed most to the paper. As this is closely connected with the most productive (the academically 'oldest' author), we also compute the success score for this task.

We study the success rate of our framework for the three different tasks using the SVM classifier with writing style features. The results of this experiment are depicted in Figure 4 (a). The x-axis of this plot is the rank of the candidate authors provided by the classifier. As the attacker slavishly follows this ranking when carrying out his guessing attack, this axis can be interpreted as the number of guesses of the attacker. The y-axis is the probability, i.e., fraction of papers in the test set, where the respective task is solved with that many guesses. For instance, for 23% of all papers, the first guess of this classifier suffices to predict the most productive author, i.e., to solve the task *guess-most-productive-one*. To solve the same task for 47% of the papers ten or more guesses are needed.

As we expected, the prediction accuracy of the classifier on all tasks increases quickly with the number of guesses. The classifier achieves the best prediction accuracy for the *guess-one* task, with around 28% success rate for the first guess, i.e., 28% accuracy that the top ranked candidate is among the true authors. The worst prediction accuracy is achieved on the *guess-all* task. Interestingly, the prediction accuracy (around 23%) for *guess-most-productive-one* is close to that of *guess-one*. This confirms our belief that the most productive author is the one that is the easiest to guess.

We also evaluated the prediction accuracy for all three tasks using other classifiers with other features. In each case, we observe a very similar pattern. Based on these findings, we conclude that the *guess-one* task can be treated as the task that a real-world attacker is most likely interested in, namely correctly guessing the research group that has submitted the paper. Therefore, in what follows, we only show the prediction results for the *guess-one* scenarios.

D. Citation Data and Classifiers

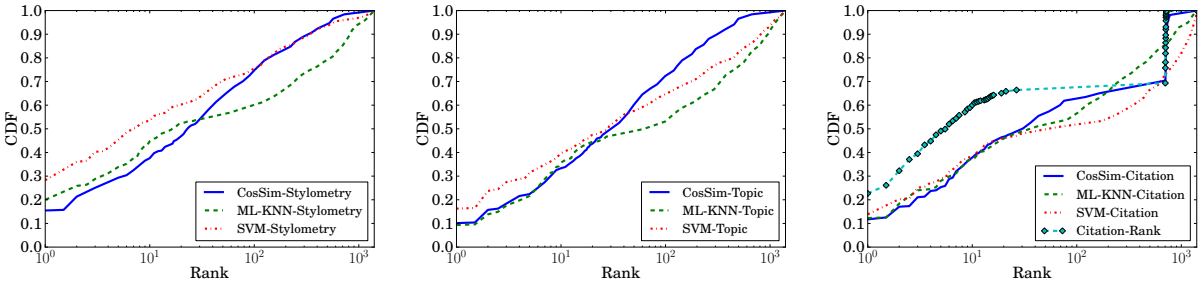
We now present an overview of the citation data we extracted from the entire corpus of all papers, followed by the evaluation of all classifiers, proposed in Section IV-B, that use citation features as an input. Later, we take the best such classifier as an input for our ensemble method.

1) Basic citation statistics: Figure 5 depicts an overview of the citation data. Figure 5 (a) shows the citation statistics over all papers in our database. For each paper, we counted the number of references for which we were able to correctly retrieve a cleaned-up version from DBLP. The histogram indicates that most papers cite between 5 and 15 other publications. While this distribution rises quickly on the left flank it has a long tail reaching to 50 or more citations. Figure 5 (b) illustrates the citation network of the authors in our data set. Entry (i, j) of this matrix indicates if author i has cited author j at least once. If so, the entry is plotted by a colored dot, if not it is white. The rows and columns of this matrix are sorted in two ways. First, authors are added conference by conference with a random ordering of conferences (the biggest conference in this matrix is NIPS). If an author has been already added by one conference he will not be added again even if she/he published at other conferences. Second, within a conference, earlier proceedings come first. This organization of the matrix highlights two dominant effects in this citation network. First, authors tend to cite authors that publish in the same conferences. Some conferences are well connected, indicating that they belong to the same scientific community. Second, the earlier an author has published, the more frequently he/she is cited on average.

2) Citation-based classification: We now compare the guessing power of different weighting functions for author rankings based on weighted cosine similarity of the citations. Recall that these methods work as follows. For each test paper, we first compute a nearest neighbor ranking by sorting all training papers by decreasing cosine similarity between their citation vector and the citation vector of the test paper. As defined in Equation 1, the score of each candidate author is a sum over all of her or his training papers, where each training paper contributes with a weight that depends on its nearest-neighbor rank with respect to the test paper. Figure 3 (a) depicts all such weighting schemes that we used.

Figure 4 (b) indicates the guessing power of all citation-based methods on the hold-out test data. It is apparent that the Citation-Rank and Top-Cited author methods work best with about 23%. Second best are the exponentially decaying weighting schemes. Thereby, the exponential decay base of 9% per paper rank ('CMU weights') and our base of 18% per paper rank selected by cross-validation ('learned weights'), perform almost equally well with a hit chance of 20%-21%. These classifiers even produce estimates when the true authors are not cited in the paper, which explains why they achieve good results for a large number of guesses (beyond 50). The linear weighting scheme performs worst with a 2% hit chance of the first guess. The generic CosSim classifier applied to the citation features achieves 12%

Overall, the best classifier on citation features is to assign each candidate author a score that equals the number of times the author is cited by the paper ('Citation-Rank'). It correctly



(a) Writing Style (b) Topic (c) Citation

Fig. 6: Performances of different classifiers with (a) writing style features, (b) topic features and (c) citation features. The y-axis is the probability that the correct author is among the first K guesses, for each K (x-axis).

guesses at least one author for 23% of all test papers and for later guesses its success rate grows faster than for all other methods. Therefore, we take this classifier output as input for our ensemble method. Taking the small set of most-cited authors ('top-cited author') performs equally well on the first and second guess. However, for third and later guesses its success rate grows significantly slower than using the citation numbers to rank these candidates.

E. Success Rates of Generic Classifiers

We investigate the predictive power of the generic classifiers with different features. We generate guessing attacks with each of the three machine learning methods (CosSim, ML-kNN and SVM) with each of the three different types of features (writing style, topic, and citation features), and show their prediction accuracy in Figure 6. For citation features, we also use the method of weighting each candidate author by how often he or she is cited by the respective paper. We observe for all classifiers on all features that the prediction accuracy increases quickly with the rank. From Figure 6 (a) we see that with the writing style features, the SVM classifier achieves the best prediction accuracy up to rank 208, and then is dominated by the CosSim classifier. From Figure 6 (b) we see that with topic features, SVM classifier dominates the other two when the rank is less than 6, and CosSim classifier dominates when the rank is larger than 30. ML-kNN always performs the worst. From Figure 6 (c) we see that with citation features, all three generic classifiers perform similarly. The Citation-Rank classifier, the method of ranking each candidate by the number of citations, has significantly higher chances to succeed across almost all ranks. For this method we see a sharp jump of performance after rank 500 and 70% success chance. This point marks the fraction of the remaining 30% of the papers that do not cite any of the true authors of the paper.

Across plots (a), (b) and (c) in Figure 6, we observe that: first, in a low rank region the Writing Style classifier performs the best (e.g., with around 29% accuracy for the first guess, i.e., the top ranked candidate is among the true authors), Citation-Rank classifier performs the second best (e.g., with around 23% accuracy for the first guess). Second, the topic features have weaker prediction power than the other two features, with SVM classifier achieving about 18% accuracy for the first guess.

F. Ensemble Classifier

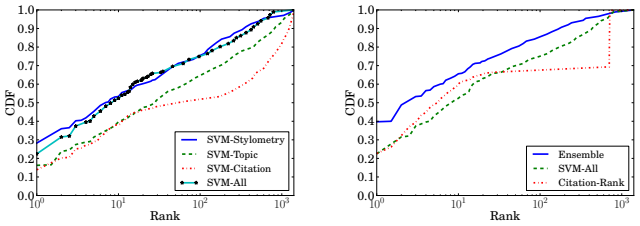
We evaluate methods that combine all classifiers with all the features together to achieve better accuracy for author identification.

Feature concatenation is suboptimal: As depicted in Figure 6, different features have different predictive power for author identification, and different classifiers have different capabilities to extract relevant information from features that is informative with respect to authorship. Therefore, we do not expect a single method to achieve the best results in combining all the features. Instead we expect an ensemble of several different classifiers on the features to achieve best results.

To prove our hypothesis, we first study how a single classifier performs on data that simply concatenates all the features together, i.e., for each paper, its feature representation includes all features from writing style, topic, and citation. In this experiment, we normalize the features from different sets in a different way, train an SVM classifier on each of the possible normalization settings, and plot the best prediction results in Figure 7 a). We see that the performance of SVM with all features concatenated is very close to or even worse for small ranks than that of SVM with only writing style features. We also train and test CosSim and ML-kNN classifiers on the concatenated feature data. The results of all these methods are almost the same. This phenomenon partially results from the fact that for each paper, both the topic features and citation features are far more sparse than the writing style features. When we concatenate these features, writing style feature data dominates the other two, and the other two features contribute far less to the accuracy of authorship prediction.

G. Ensemble method versus state-of-the-art

We now investigate the performance of our ensemble method and compare it with the feature-concatenation method and Citation-Rank classifier, which is the state-of-the-art technique for predicting authors of scientific papers. Among the 10 classifiers (CosSim, ML-kNN and SVM on writing style, topic and citation, respectively, plus the Citation-Rank classifier), we observe from Figure 6 (c) that the Citation-Rank performs significantly better than the three generic citation classifiers, so we decide to drop the three classifiers and only use



(a) Feature concatenation. (b) Comparing to previous work.

Fig. 7: a) Feature concatenation does not work. b) Comparing our ensemble methods to previous work.

	Writing style	Topic	Citation
SVM	0.277	0.121	/
CosSim	0.277	0.000	/
ML-kNN	0.091	0.000	/
Citation-Rank	/	/	0.234

TABLE III: Weights of different classifiers in the ensemble model for the year 2012.

the remaining seven classifiers for ensemble learning, whose results are shown in Figure 7 b) and Table III.

From Figure 7 b), we see that our ensemble method significantly outperforms both other methods, especially for the low-rank region, where the attacker tries to pin down the authors by a small number of guesses. More precisely, our method achieves around 39.7% accuracy for a single guess, and a 65.6% chance that 10 guesses contain at least one true author. The other methods both have 22.5% (SVM-All) or 23% (Citation-Rank) success chance with the first guess and a 52.4% (SVM-All) or 60.4% (Citation-Rank) chance with 10 guesses. This result confirms that combining multiple modalities in a consistent way extracts more discriminative information than using individual features or simply concatenating all features.

Table III shows the optimal weights learned by our ensemble method for combining different classifiers together. We observe the following two interesting results. First, large weights are allocated to classifiers trained on both writing style features (column 1) and citation features (column 3), which confirm our intuition that they are predictive features and complementary to each other. Second, zero or small weights are allocated to classifiers trained on topic features (column 2). Our hypothesis is that although topic features are predictive, they are redundant to and dominated by citation features, so the ensemble does not bother to use them. To test our hypothesis, we train an ensemble classifier using the six classifiers derived from writing style features and topic features (but not citation features). As shown in Table IV, this ensemble classifier puts much more weight on classifiers trained on topic features, although the prediction accuracy degrades (e.g., single-guess accuracy degrades to 26.1% from 39.7%), due to the classifier not using citation features.

H. Cross-validation

The different contents of the papers in our dataset, the citations, the words used, and the scientific topics are all

	Writing style	Topic	Citation
SVM	0.343	0.258	/
CosSim	0.183	0.086	/
ML-kNN	0.129	0.000	/
Citation-Rank	/	/	/

TABLE IV: Weights of different classifiers in the ensemble model without citation features for year 2012.

	Guess-one	Guess-10	# train. papers	# authors
2010	43.2%	72.6%	2401	969
2011	39.0%	72.0%	2964	1184
2012	39.7%	65.6%	3516	1405

TABLE V: The cross-validation results for the ensemble classifiers in years 2010, 2011, 2012.

random variables following some unknown distributions. As a consequence the features that we extract from this data and consequently also the predictions of our framework and the resulting performance scores are random variables. As the number of observations (here the number of papers) is finite, the valid question arises to what extent our results vary with different random realizations of the involved random variables.

In order to investigate the significance of our findings we have carried out a cross-validation experiment where we run our framework on different subsets of the data-set. This approach is commonly used to evaluate methods that operate on finite observations of random variables. Usually one simply subdivides the given dataset into several random subsets and carries out the method on these subsets such that different quantiles of the target scores can be computed. Our particular problem has two aspects that render traditional cross-validation challenging. First, holding out papers from the training set reduces the number of papers for some authors below three papers such that they might drop out as candidate authors. As the number of candidates influences the difficulty of the deanonymization problem, it is difficult to compare across random subsets of papers. Even worse, even though our training dataset consists of many papers, only a small subset of authors have more than three papers. This means reducing the number of papers by subdividing the training set can have a big effect. Second, papers appear in a chronological order that cannot be ignored. In particular, younger papers cite older papers and older papers cannot cite younger papers. Also, scientific topics emerge at one point in time and there are no papers about that topic before that time.

Considering these challenges, we have carried out three different experiments for three consecutive years from 2010 to 2012. For each year we took all papers prior to that year as the training set and the papers of that year as the test set. This means there are no chronological inconsistencies. However, there are fewer training papers and fewer candidate authors for 2010 than for 2012. For earlier years, this effect would be so significant that we did not go further back in time.

The results are depicted in Table V. We report the number of training papers and the number of candidate authors along with two success rates of guessing authors: correctly guessing with one guess and hitting at least one correct author with the first

10 guesses. Clearly, fewer training observations (fewer papers) render the classifiers weaker. In contrast, fewer candidate authors make it easier to guess the true authors by chance. Despite these factors, the success rates vary in terms of a few percents only. Taking the trend of decreasing accuracies into account, it seems like the number of candidate authors is the dominating factor that overrides the statistical variations as well as the influence of the number of training papers.

I. Processing Time

Our deanonymization pipeline consists of three stages: paper parsing, feature extraction, and classifier training. We report the processing time used in each stage, and discuss feasible ways to further speedup the pipeline. Unless otherwise stated, performance numbers are derived by running our pipeline on a single core on an Intel Xeon X5550 machine at 2.67GHz.

1) Paper parsing: Extracting all text elements and layout reconstruction of a paper takes less than 30s on average. Matching all references against a data set of well-known clean references takes 10-30s per reference, resulting in an overall processing time of 2 to 10 min. per paper. Data extraction is embarassingly parallel as each paper can be processed independently.

2) Feature extraction: Our pipeline extracts writing style (1206s), topic (3745s) and citation network features (273s) in parallel for all papers. Topic feature extraction dominates this stage. The majority of time is used for stop-word removal and word stemming, which can be parallelized linearly (e.g., using 4 cores, the processing time is reduced to 839s). Fortunately, we only need to do this vocabulary construction once for all papers.

3) Classifier training: The classifier training stage consists of two steps: training individual classifiers and ensemble learning. We use seven individual classifiers for the ensemble, and all of them can be trained in parallel because there is no dependence between different classifiers. So the processing time in this step is dominated by the classifier taking the longest time. Among the seven classifiers, ML-kNN with writing style features takes the longest time (1898s). A large portion of the time is spent for the computing of the pair-wise similarity, which can be easily sped up using a scalable similarity search algorithm [3]. SVM training for content features ranks second (1562s) as we build one classifier per author. Classifiers for individual authors could be trained in parallel.

Once individual classifiers are trained, the ensemble learning is fast: it takes around 133s to search for the optimal weights to combine different classifiers together.

4) Processing summary: It only takes 2 to 10 minutes of initial processing per paper. This paper parsing part must only be executed once per paper. It takes less than 1 hour to extract the features and to train the classifiers. This step must be repeated whenever new data is added to the training corpus. It takes between 2 to 10 minutes to attack a new, anonymized paper given a pre-trained data-set.

VI. PDF Container Leaked Data Features

During PDF compilation, many PDF generators add auxiliary information about embedded files to the PDF. This includes which tools and versions were used to generate figures or tables, the directories of the figures relative to the paper source, the internal original name of the citation in the LaTeX file, or even the user name or company name, which holds the program license. This information is only infrequently available and very diverse such that it is hard to include it to a large-scale approach like ours. However, it can easily complement our attack as it can discard or highlight authors in our ranking.

Out of 5,477 original papers 4,766 contained some form of leaky strings. 532 (11.2%) papers directly contained a total of 1,600 author names that make it trivial to identify the original authors. This hidden author string is orthogonal to the clear text in the title section. Browsing through the list, we discovered many email addresses and full names that might have been used during the registration of software components. Some PDF tools extract the username and the associated full name of the user that is producing the PDF and automatically embed it in the PDF. In addition, we found 7 papers that included a company name (embedded in licensing information for software components used to generate content).

When looking at the creator and producer options in the raw PDF stream we discovered that 1009 papers (21.2%) contain at least one leaky creator and 1127 (23.4%) papers contain a leaky producer. Each paper with a creator contains on average 4.0 creators and each paper with a producer contains on average 2.9 producers. Creators and producers identify the software (and often the exact version and the operating system kernel) that was used to generate the PDF or a figure of the PDF. In addition, 255 papers (5.4%) contained a total of 2219 original file names for embedded figures and 9 papers contained absolute file names that leak the user name of the person generating the PDF.

If this information is available in an anonymous submission, then it is a strong indicator for the original author. However, since this information can easily be *removed* (e.g., the surplus information like user names) or *randomized* (e.g., for internal citation references), we refrain from leveraging any of those features in our evaluation and discuss them here only for completeness. We urge authors (and tool writers) to remove this information for anonymous submissions.

VII. Discussion

We have seen that our framework outperforms state-of-the art author deanonymization. In this section we discuss practical aspects of deploying our method in a conference submission system. The offline approach presented in this paper uses only published proceedings of earlier conferences for the analysis and openly published and accessible DBLP data to verify the extracted references. Using this data corpus a reviewer can attack the anonymity of a submitted paper and recover the authors possibly using additional online information like a web search.

A. Automatic anonymity checker

A system that breaks author anonymity can be used in an online setting as part of a conference submission system.

Comparable to a format checker, the submission system automatically parses any uploaded paper and checks if anonymity holds. If any of the real authors is in the (e.g., top 10) set of projected authors then the submission system gives feedback on which features were used to deanonymize the paper (e.g., specific citations that stick out, writing style of *specific* sections, or certain topic features). The authors can then change the paragraphs or citations that leak most information about their identity in an iterative process.

Such an approach is feasible, can be implemented with low overhead, and can be added to large conference submission systems like EasyChair. The amount of new papers is finite and can be processed when released to the digital libraries. Each paper must be parsed once (2 to 10 min. per paper) and the classifiers need to be trained after adding new proceedings to the training set (up to a few hours of computation). In the online setting a testing paper is parsed in roughly 2 min. and can then be matched against the existing classifiers in seconds.

B. Mitigation

Looking at the ensemble set up, we see that the citation classification and writing style features help most in identifying possible authors. While it is hard to consciously change one's writing style it is easier to decrease the prediction quality of the citation classifier.

Given that the number of citations per candidate author is the best individual classifiers, one should not use too many self-citations. Also, many authors cite the same (sub-)set of papers for different papers. The citation classifier identifies these sub-sets and uses them to match different authors. So as a second mitigation strategy authors should minimize the shared set of cited papers to the necessary related work.

C. Additional features

The format of a paper can also leak information about authorship as different authors format and structure papers differently. The location of the related work section (beginning or end), the placement of figures on the page (top, bottom, left or right column), the length of figure or table captions, the length of the title or the individual sections can all reveal information about the author.

The relationship between sets of authors evolves over time as well as authors shift their focus area and they publish on different topics. Temporal features group authors and co-authors according to temporal information. For example, it is much more likely that an author worked on a current paper together with a recent co-author, than with a co-author he or she stopped working together several years ago. The same argument also holds for topics: an author is more likely to publish in a related area where he or she worked recently in, rather than in an area where the author stopped publishing years ago.

While in our current model we do not use these additional features, our prediction engine is open for additional feature sets and classifiers, but we leave adding paper format features and temporal features for future work.

VIII. Related Work

Much prior work studies the degree of anonymity of the review process for scholarly research articles. Nanavati et al. [26] show that stylometry enables identifying reviewers of research papers with reasonably high accuracy, given that the adversary has access to a large number of unblinded reviews of potential reviewers by serving on conference and grant selection committees. Several researchers have studied the feasibility of identifying the author of an academic paper under blind review solely from the citations [15], [5]. Our work goes one step further to demonstrate that by gracefully combining writing style features, topic features, and citations, identification accuracy is greatly improved.

There is a long list of prior work on identifying the author of a text based on the writing style. The seminal work of Mosteller and Wallace [25] leverages function words and Bayesian analysis to identify the authors of the disputed Federalist Papers. Research in the last decade has focused on the machine-learning paradigm and the inclusive approaches to feature extraction [21], [1], [11]. These studies consider "topic-free" models and are able to discriminate between 100-300 authors. Afroz et al. [2] study author identification in adversarial scenarios, and propose an effective method for detecting stylistic deception in written documents.

Several other recent approaches extend the existing methods to large-scale author identification. Koppel et al. [18], [17] study authorship recognition on a blog corpus spanning 10,000 authors. Their work makes use of both content-based features and stylistic features to identify authorship with certain success rates. To remove the interference from context, Narayanan et al. [27] perform a large-scale study on the feasibility of Internet-scale author identification using stylistic features only, and achieve 20% accuracy on a corpus of texts from 100,000 authors.

Much research has been carried out to investigate techniques for transforming text to successfully resist author identification [6], [23]. These papers consider off-the-shelf stylometry attacks and propose semi-automated techniques to identify where and how to change the document to accomplish its anonymization.

IX. Conclusion

We presented deAnon, a framework that solves the Paper Deanonymization Problem using a multi-label, multi-class machine learning approach. Based on a large data corpus of existing proceedings from a diverse set of conferences, deAnon trains 1,405 per-author classifiers based on multiple heterogeneous modalities like their writing style, published topics, and their citation behavior. On queries of anonymized papers, deAnon returns a ranking of candidate authors.

Further, we evaluate deAnon using proceedings from 17 computer science conferences from 1996 to 2012 with 3,894 total papers, splitting these submissions into train and test data sets. deAnon recovers one author with 39.7% probability on the first guess and with 65.6% probability the top-ten guesses contain at least one true author, significantly outperforming

prior work (less than 21% for the first guess using the CMU classifier on our data set).

These results demonstrate that deanonymization of anonymous paper submissions is feasible and confirms the common belief shared by many reviewers. Anonymity is considerably limited by the effectiveness of authors to change their behavior: either authors of anonymous submissions need to take more care to anonymize their papers or we as a community need to rethink the anonymous review process.

REFERENCES

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

[2] AFROZ, S., BRENNAN, M., AND GREENSTADT, R. Detecting hoaxes, frauds, and deception in writing style online. In *IEEE Symposium on Security and Privacy* (2012), IEEE Computer Society, pp. 461–475.

[3] BAYARDO, R. J., MA, Y., AND SRIKANT, R. Scaling up all-pairs similarity search. In *Proceedings of WWW* (2007).

[4] BLEI, D., NG, A., AND JORDAN, M. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003).

[5] BRADLEY, J. K., KELLEY, P. G., AND ROTH, A. Author identification from citations. *Technical Report, CMU* (Dec. 2008).

[6] BRENNAN, M., AND GREENSTADT, R. Practical attacks against authorship recognition techniques. In *IAAI* (2009).

[7] BROWN, G., WYATT, J., HARRIS, R., AND YAO, X. Diversity creation methods: a survey and categorisation. *Information Fusion 6*, 1 (2005).

[8] CARTER, R., GABLONSKY, J., PATRICK, A., KELLEY, C., AND ESLINGER, O. lgorithms for noisy problems in gas transmission pipeline optimization. *Optimization and Engineering 2* (2002).

[9] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011).

[10] CORTES, C., AND VAPNIK, V. Support-vector networks. *Journal of Machine Learning 20*, 3 (1995), 273–297.

[11] DE VEL, O. Y., ANDERSON, A., CORNEY, M., AND MOHAY, G. M. Mining email content for author identification forensics. *SIGMOD Record 30*, 4 (2001), 55–64.

[12] DZEROSKI, S., AND ZENKO, B. Is combining classifiers with stacking better than selecting the best one? *Machine Learning 54*, 3 (2004).

[13] ET AL., H.-F. Y. Feature engineering and classifier ensemble for kdd cup 2010. In *KDD Cup* (2010).

[14] FARIDANI, S. An nlp library for matlab. *https://github.com/faridani/MatlabNLP*.

[15] HILL, S., AND PROVOST, F. The myth of the double-blind review?: author identification using only citations. *SIGKDD Explor. Newsl. 5*, 2 (Dec. 2003), 179–184.

[16] JONES, D., AND PERTTUNEN, C. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application 79*, 1 (1993).

[17] KOPPEL, M., SCHLER, J., AND ARGAMON, S. Computational methods in authorship attribution. *JASIST 60*, 1 (2009), 9–26.

[18] KOPPEL, M., SCHLER, J., AND ARGAMON, S. Authorship attribution in the wild. *Language Resources and Evaluation 45*, 1 (2011), 83–94.

[19] LEWIS, D. D., YANG, Y., ROSE, T. G., AND LI, F. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* (2004).

[20] LEY, M., AND REUTHER, P. Maintaining an online bibliographical database: The problem of data quality. In *EGC* (2006), pp. 5–10.

[21] MADIGAN, D., GENKIN, A., LEWIS, D. D., ARGAMON, S., FRADKIN, D., AND YE, L. Author identification on the large scale. In *Joint Meeting of the Interface and Classification Society of North America* (2005).

[22] MADJAROVA, G., KOCEVB, D., GJORGJEVIKJA, D., AND DZEROSKIB, S. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition 45*, 9 (2012).

[23] MCDONALD, A. W. E., AFROZ, S., CALISKAN, A., STOLERMAN, A., AND GREENSTADT, R. Use fewer instances of the letter "i": Toward writing style anonymization. In *Privacy Enhancing Technologies* (2012), pp. 299–318.

[24] MENDENHALL, T. C. The characteristic curves of composition. *Science ns-9*, 214S (1887), 237–246.

[25] MOSTELLER, F., AND WALLACE, D. L. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, 1964.

[26] NANAVATI, M., TAYLOR, N., AIELLO, W., AND WARFIELD, A. Herbert west: deanonymizer. In *HotSec* (2011).

[27] NARAYANAN, A., PASKOV, H., GONG, N., BETHENCOURT, J., STEFANOV, E., SHIN, E., AND SONG, D. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy* (may 2012), pp. 300 –314.

[28] PAPINENI, K. Why inverse document frequency? In *Proceedings of NAACL* (2001).

[29] ROKACH, L. Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1-2 (2010).

[30] ROSEN-ZVI, M., GRIFFITHS, T., STEYVERS, M., AND SMYTH, P. The author-topic model for authors and documents. In *Proceedings of UAI* (2004).

[31] SCULLEY, D., AND WACHMAN, G. M. Relaxed online svms for spam filtering. In *Proceedings of SIGIR* (2007).

[32] SHANTANU GODBOLE, S. S. Discriminative methods for multi-labeled classification. In *PAKDD* (2004).

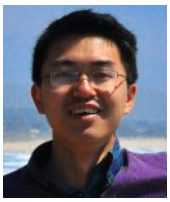
[33] SPYROMITROS, E., TSOUMAKAS, G., AND VLAHAVAS, I. An empirical study of lazy multilabel classification algorithm. In *Proceedings of Hellenic conference on Artificial Intelligence* (2008).

[34] ZHANG, M.-L., AND ZHOU, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition 40*, 7 (2007), 2038 – 2048.

[35] ZHU, H., AND BOGY, D. Direct algorithm and its application to slider air-bearing surface optimization. *IEEE Transactions on Magnetics 38*, 5 (2002).

Mathias Payer is a security researcher and an assistant professor in computer science at Purdue university. His interests are related to system security. Before joining Purdue in 2014 he spent two years as PostDoc in Dawn Song's BitBlaze group at UC Berkeley. He graduated from ETH with a Dr. sc. ETH in 2012.

Ling Huang is a founding member and member of technical staff of Datavisor Inc. He was a research scientist in Intel Labs Berkeley from October 2007 to May 2014. His research interests are in machine learning, systems and security. Ling Huang joined Intel Labs Berkeley in October 2007 after finishing his Ph.D. in CS at UC Berkeley.



Neil Zhenqiang Gong is a Ph.D. candidate in Computer Science at the University of California at Berkeley. Prior to Berkeley, he studied computer science at the University of Science and Technology of China (USTC) and obtained B.E. in computer science in 2010. He is interested in security, privacy, and their intersection with data science.

Kevin Borgolte is a Ph.D student in Computer Science at the University of California, Santa Barbara. Prior to joining the Ph.D. program, he received his Master's degree from ETH Zurich, Switzerland in 2012 and his Bachelor's degree from the University of Bonn, Germany in 2010. His research interests span web, software and systems security, and the applications of machine learning and natural language processing in information security and privacy.

Mario Frank is an examiner at the European Patent Office (EPO). Prior to joining the EPO, he was with the information security groups of UC Berkeley and ETH Zurich and has studied physics at the Ruprecht-Karls-University Heidelberg and the University of Sydney. He has a broad interest in problems at the intersection of machine learning and information security.