


Article

First Steps towards Data-Driven Adversarial Deduplication

Jose N. Paredes ^{1,2,†}, Gerardo I. Simari ^{1,2,3,†,*} , Maria Vanina Martinez ^{4,5,†}
and Marcelo A. Falappa ^{1,2,†}

¹ Department of Computer Science and Engineering, Universidad Nacional del Sur (UNS), 8000 Bahia Blanca, Argentina; jose.paredes@cs.uns.edu.ar (J.N.P.); mfalappa@cs.uns.edu.ar (M.A.F.)

² Institute for Computer Science and Engineering (CONICET-UNS), 8000 Bahia Blanca, Argentina

³ School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University, Tempe, AZ 85281, USA

⁴ Department of Computer Science, Universidad de Buenos Aires (UBA), C1428EGA Ciudad Autonoma de Buenos Aires, Argentina; mvmartinez@dc.uba.ar

⁵ Institute for Computer Science Research (CONICET-UBA), C1428EGA Ciudad Autonoma de Buenos Aires, Argentina

* Correspondence: gis@cs.uns.edu.ar; Tel.: +54-291-459-5101 (ext. 2628)

† These authors contributed equally to this work.

Received: 26 June 2018; Accepted: 23 July 2018; Published: 27 July 2018

Abstract: In traditional databases, the entity resolution problem (which is also known as deduplication) refers to the task of mapping multiple manifestations of virtual objects to their corresponding real-world entities. When addressing this problem, in both theory and practice, it is widely assumed that such sets of virtual objects appear as the result of clerical errors, transliterations, missing or updated attributes, abbreviations, and so forth. In this paper, we address this problem under the assumption that this situation is caused by malicious actors operating in domains in which they do not wish to be identified, such as hacker forums and markets in which the participants are motivated to remain semi-anonymous (though they wish to keep their true identities secret, they find it useful for customers to identify their products and services). We are therefore in the presence of a different, and even more challenging, problem that we refer to as adversarial deduplication. In this paper, we study this problem via examples that arise from real-world data on malicious hacker forums and markets arising from collaborations with a cyber threat intelligence company focusing on understanding this kind of behavior. We argue that it is very difficult—if not impossible—to find ground truth data on which to build solutions to this problem, and develop a set of preliminary experiments based on training machine learning classifiers that leverage text analysis to detect potential cases of duplicate entities. Our results are encouraging as a first step towards building tools that human analysts can use to enhance their capabilities towards fighting cyber threats.

Keywords: adversarial deduplication; machine learning classifiers; cyber threat intelligence

1. Introduction and Motivation

The classical problem of *entity resolution*—or *deduplication*—in databases seeks to address situations in which seemingly distinct records are stored that actually refer to the same entity (object, person, place, etc.) in the real world. Typically, the goal is to identify and merge such records [1,2]. See Section 4 for a discussion of related work.

The characteristic that is overwhelmingly shared among these traditional approaches is that they assume that the existence of multiple records for the same real entity is the product of *involuntary* situations such as simple typos during data entry procedures, ambiguity in attribute values such as

transliterations and abbreviations, and inconsistency and incompleteness due to overspecification and underspecification (two addresses for the same person, or address completely missing), respectively, or evolving values such as address changes. In this paper, we are interested in situations in which these assumptions simply cannot be made because there are actors who may purposefully be taking actions towards hiding their identity behind multiple profiles. Take, for instance, the setting of malicious hacker forums on the Dark/Deep Web, in which participants seek to buy and sell different kinds of goods such as malware, passwords, credit card numbers, and other illicit materials. There is an interesting dynamic that arises among the participants in these forums and marketplaces: though of course they wish to remain anonymous—especially from government agents who may be watching—they on the other hand also wish to maintain their reputation within the community, and must therefore remain identifiable. The same actor typically operates using different profiles, but keeping certain characteristics constant. Perhaps most importantly, they also leave *involuntary traces* behind that can be analyzed and leveraged by deduplication tools. We refer to this as the *adversarial deduplication/entity resolution problem*. Figure 1 illustrates this situation via a simple visualization. Consider the problem of trying to determine clues that point to the conclusion that a given pair of faces might correspond to the same real-world user (or perhaps to the opposite conclusion).

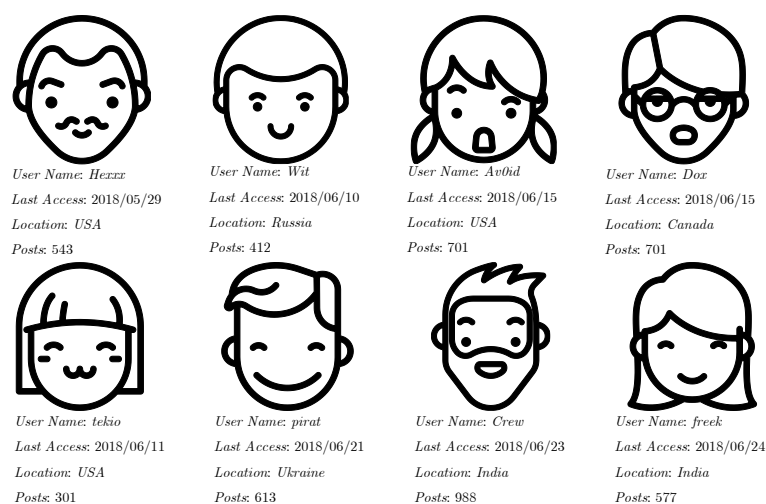


Figure 1. In real-world applications such as Dark Web forums and marketplaces, it may not always be clear who is behind user profiles. In particular, there may be two or more profiles that correspond to the same person.

The following is a simplified example of the kind of information that we can obtain from dark web forums and marketplaces.

Example 1. Consider the database schemas shown for the tables in Figure 2, where we have a table for forums, topics, and posts. In this paper, we focus on the posts, since they are the main source of material that can be used towards identifying potential duplicates.

This information is based on the system developed in [3] and CYR3CON (<https://cyr3con.ai>) for cyber threat intelligence, which scrapes data from various social platforms, especially in the dark net and deep net. They collect and store information from hacker forum discussions and marketplaces offering products and services that focus on malicious hacking, such as sales of malware/exploits (including CVE numbers, which are identifiers given by the National Vulnerability Database [4,5]) and hacker forums (discussions regarding services and threats). The crawling and parsing of these sites yields time-varying data, since the system periodically returns to the same sites.

postId	postContent	postedDate	scrapedDate	forumId	recordedDate	userId	topicId
11e9d297a29899f6a9c3da05391acaa	[u', the only good way to work in computer security straight is to be a "black hat" then at some point (arrested a few to many times, you get a wife/family) you decide you can't do it any more. the other way is to go into sysadmin and at some point, once you have experience going into nothing but security. . . u", i see way to many people about trying to sell themselves as pen-testers and security consultants and know fuck all, don't be one of those people., u"]	11/10/2008	1/22/2017	56	11/10/2008	352820	481581
da1a4e396af0e1b87ba0d3a81b1e6277	your only recourse is using a second 3.60 or updated unit, or ps3 to download the games, then transfer them to your vita or qcma. and that only works for games you purchased, it isn't a gateway to piracy.	6/3/2017	6/5/2017	134	6/3/2017	75065	821237
92ac4e1e3cdb886080c6af2a528673eb	and i don't particularly agree with touting this method either. why use wm_vsh_menu to call wmm functions to prepare the cfw when pspatch does a fine job...?it's not explained here but i assume it's to run jb folder games to go online. however there is a good reason for pspatch to remove cobra hooks stopping many jb folder games from working. bypassing this feature is not a smart idea for most users. i don't recommend to do it. cfw users should always stick to iso & the pspatch method. as to the title given the fact that the only users who require jb folders are cheaters & modders i wouldn't actually hold my breath when telling them they will "never" get banned if they follow these steps... assuming that these steps will prevent a ban is not the same as knowing they will....	3/19/2017	6/23/2017	93	3/19/2017	1807	946623

forumId	boardsName
56	null
134	vitahacks
93	thread locked

topicId	topicName
481581	[u'white hat hacker carrer']
821237	bought psn games on 3.63question (self.vitahacks)
946623	thread locked

Figure 2. Snippets of the *hackingPosts*, *forums*, and *topics* tables from our dataset.

The rest of this paper is organized as follows: Section 2 presents our method to train machine learning classifiers to recognize posts made by users based on text analysis techniques, Section 3 presents our empirical evaluation consisting of two main experiments (a first phase consisting of a broad evaluation of different classifiers and hyperparameter settings, and a second phase analyzing the effectiveness of the best two in finding pairs of entities), and Sections 4 and 5 discuss related work and conclusions, respectively.

2. Deduplication Leveraging Text-Based Features

We now present a proposal for applying machine learning techniques towards solving adversarial deduplication problems. Note that the general approach is not novel, and has been applied in several other problems, such as malware identification and attribution, among others (cf. Section 4 for a discussion). However, to the best of our knowledge this is the first such proposal for this problem. Essentially, we wish to develop a *lightweight* method by which posts written by users can be automatically analyzed and *deduplication hypotheses* can be generated so that human analysts can step in to provide a more in-depth analysis. The workflow can be summarized in the following steps:

- Procure information from online discussions in forums and marketplaces; this is an ongoing effort that is generally carried out in a semi-automatic manner by specialists [6].
- Prepare the data by performing several cleaning processes (see below).
- Train one or more machine learning classifiers to recognize posts written by each user. For this step, we assume that posts made under different user names correspond to different users—they come back to this assumption when analyzing the results yielded in the testing phase. Note that another option is to train a single multi-class classifier that discriminates among all users. We decided to train one classifier per user for two main reasons: (a) large numbers of users (classes, in that case) are more difficult to manage, and the resulting classifier would be less flexible—by training one classifier per user it is possible to incorporate features that only apply to certain users; (b) perhaps most importantly, we would like to be flexible with respect to the incorporation of new users, which would cause the single classifier to be retrained with each addition.
- Apply the classifiers to *pairs* of new posts by users *X* and *Y*; if either *X*'s classifier states that a post written by *Y* was written by *X*, or vice versa, then we generate a *deduplication hypothesis*.
- The set of deduplication hypotheses are sent to human analysts for further treatment.

Note that if we have no further information about the authors of posts, for n users we would have to analyze $\binom{n}{2}$ pairs to see if they actually correspond to the same user. For 50 users, this amounts to 1225, and for 100 we have 4950, which are already intractable numbers. Clearly, scaling such a brute force analysis to larger numbers of users is impossible (cf. Table 1). The general goal of our method is therefore to greatly reduce the set of pairs that humans must actually look at.

Table 1. Numbers of unordered pairs of users needed to be inspected by brute force analysis.

Number of Users n	Number of Possible Pairs $\binom{n}{2}$
50	1225
100	4950
150	11,175
200	19,900
500	124,750
1000	499,500
2000	1,999,000
5000	12,497,500
10,000	49,995,000

Analyzing text via n -grams: In order to extract basic elements from text, one common tool is the use of an n -gram, which can be defined in different ways. Here, we adopt the commonly used definition of an n -gram as a sequence of n characters. The advantage of using n -grams instead of directly analyzing a text is that typos, spelling variations, and other kinds of differences yield sets of n -grams that are closely related.

Example 2. Consider the word “software” and some common misspellings/intentional variations used by online communities: *software*, *softwarez*, and *sophwarez*. If we consider the 2-grams and 3-grams for each of these terms, we arrive at the following sets:

	2-grams	3-grams
<i>software</i>	so, of, ft, tw, wa, ar, re	sof, oft, ftw, twa, war, are
<i>software</i>	so, of, fw, wa, ar, re	sof, ofw, fwa, war, are
<i>softwarez</i>	so, of, ft, tw, wa, ar, re, ez	sof, oft, ftw, twa, war, are, rez
<i>sophwarez</i>	so, op, ph, hw, wa, ar, re, ez	sop, oph, phw, hwa, war, are, rez

Clearly, even the two most different variations (*software* and *sophwarez*) still share several n -grams. The value of n is a parameter to be tuned—a range within [3,7] has been found to work well in this kind of analysis [3].

Our working hypothesis is therefore that properly trained machine learning classifiers can detect unintentional traces left behind in the writing of people who are trying to hide behind multiple profiles in online forums and markets. In the next section we present the design and results of a set of experiments carried out as a preliminary attempt towards proving this hypothesis.

3. Empirical Evaluation

We adopted the following basic setup for the evaluation of our approach with real-world data, which was carried out in two main experiments (see below):

- *Dataset:* posts table, which contains 89,766 posts users table, which contains 128 users.
- *Data cleaning and preparation:* We removed HTML tags from posts using the *BeautifulSoup* tool (<https://www.crummy.com/software/BeautifulSoup>), removed URLs, extra spaces, and strings that contained a combination of letters and numbers. Finally, we discarded posts that either contained less than 140 characters (i.e., anything shorter than the maximum length of an SMS

message or tweet before the expansion) or any of the following strings “quote from:”, “quote:”, “wrote:”, “originally posted by”, “re:”, or “begin pgp message”. This yielded 40,453 “clean” posts corresponding to 54 users.

- *Feature generation*: We used the well-known TF-IDF (term frequency-inverse document frequency) technique to produce vectors of features based on n -grams, which essentially consists of assigning weights to features in such a way that they increase proportionally to the number of times they occur in a document, and also takes into account the number of times the feature occurs in the whole corpus.
- *Classifiers*: Different standard machine learning approaches were implemented with a standard Python library (<http://scikit-learn.org/stable>):
 - Decision Trees
 - Logistic Regression
 - Multinomial Bayesian Networks
 - Random Forests
 - Support Vector Machines, with both linear and radial basis function (rbf) kernels
- *Hyperparameters*: We explored different values of two main hyperparameters: max_df and n_gram range. The former is a bound on the frequency with which a feature occurs in a post (essentially, as frequency increases the information content of a feature becomes lower), while the latter determines the length of the substrings into which the text is split. This yielded a set of 52 classifier instances (cf. Table 2).

For some classifier instances, we also applied a bound on the number of features taken into account by the classifier ($max_features$). This space was explored manually by testing the effect of different settings on performance and running time (more features yield better performance, up to a certain point). The final selection yielded the following:

- DT2 is DT1 with $max_features = 2500$
- DT3 is DT1 with $max_features = 2000$
- DT9 is DT8 with $max_features = 3000$
- DT5 has $max_features = 3000$
- DT6 has $max_features = 2000$
- DT18 is DT17 with $max_features = 5000$
- MNB6 is MNB5 with $max_features = 3000$

This set of classifier instances was generated by means of manual exploration of the hyperparameters, looking for the combinations that had the most potential to yield high values for precision and recall.

3.1. Empirical Evaluation Phase 1: Broad Evaluation of Different Classifiers and Hyperparameter Settings

The goal of the first set of experiments was to find the best-performing classifier instances among the 52 shown in Table 2. Towards this end, we conducted three trials of the following set of steps:

- Choose 10 users at random.
- *Training phase*: For each user u_i , train a classifier C_i using between 200 and 500 sample posts written by them (positive examples, actual number of posts depended on availability) and the same number of posts written under other screen names (*presumed* negative examples).
- *Testing phase*: For each user, take 20 *test posts* that were not used to train any of the classifiers and query each resulting classifier. This yields a confusion matrix M where each row corresponds to a user and each column to a classifier. $M(i, j)$ contains the number of posts classified by classifier C_j as corresponding to user u_i . Note that a perfect confusion matrix in this case should have

a value of 20 along the diagonal, and values of zero in all other cells. Table 3 shows an actual instance of such a matrix.

Table 2. Complete description of hyperparameter settings for all classifier instances.

ID	Classifier Type	<i>max_df</i>	<i>n-grams</i>
SVC1	SVM–linear kernel	0.02	[3,3]
SVC2	SVM–linear kernel	0.03	[3,3]
SVC3	SVM–linear kernel	0.02	[4,4]
SVC4	SVM–linear kernel	0.01	[4,4]
SVC5	SVM–linear kernel	0.03	[5,5]
SVC6	SVM–linear kernel	0.03	[3,4]
SVC7	SVM–linear kernel	0.06	[3,4]
SVC8	SVM–rbf kernel	0.01	[3,3]
SVC9	SVM–rbf kernel	0.05	[3,3]
SVC10	SVM–rbf kernel	0.03	[3,3]
SVC11	SVM–rbf kernel	0.05	[4,4]
SVC12	SVM–rbf kernel	0.08	[4,4]
SVC13	SVM–rbf kernel	0.05	[5,5]
SVC14	SVM–rbf kernel	0.03	[5,5]
DT1	Decision Tree	0.05	[3,3]
DT2	Decision Tree	0.05	[3,3]
DT3	Decision Tree	0.05	[3,3]
DT4	Decision Tree	0.02	[3,3]
DT5	Decision Tree	0.03	[3,3]
DT6	Decision Tree	0.03	[3,3]
DT7	Decision Tree	0.009	[3,3]
DT8	Decision Tree	0.03	[4,4]
DT9	Decision Tree	0.03	[4,4]
DT11	Decision Tree	0.02	[4,4]
DT12	Decision Tree	0.05	[4,4]
DT13	Decision Tree	0.01	[4,4]
DT14	Decision Tree	0.05	[5,5]
DT15	Decision Tree	0.03	[5,5]
DT16	Decision Tree	0.02	[5,5]
DT17	Decision Tree	0.01	[5,5]
DT18	Decision Tree	0.01	[5,5]
MNB1	Multinomial Bayesian Network	0.02	[3,3]
MNB2	Multinomial Bayesian Network	0.01	[3,3]
MNB3	Multinomial Bayesian Network	0.008	[3,3]
MNB4	Multinomial Bayesian Network	0.007	[3,3]
MNB5	Multinomial Bayesian Network	0.005	[3,3]
MNB6	Multinomial Bayesian Network	0.005	[3,3]
MNB7	Multinomial Bayesian Network	0.005	[4,4]
MNB8	Multinomial Bayesian Network	0.005	[4,4]
MNB9	Multinomial Bayesian Network	0.003	[5,5]
LR1	Logistic Regression	0.03	[3,3]
LR2	Logistic Regression	0.02	[3,3]
LR3	Logistic Regression	0.03	[4,4]
LR4	Logistic Regression	0.01	[4,4]
LR5	Logistic Regression	0.03	[5,5]
LR6	Logistic Regression	0.01	[5,5]
RF1	Random Forest	0.03	[3,3]
RF2	Random Forest	0.02	[3,3]
RF3	Random Forest	0.03	[4,4]
RF4	Random Forest	0.02	[4,4]
RF5	Random Forest	0.03	[5,5]
RF6	Random Forest	0.02	[5,5]

Table 3. Example of a confusion matrix for Experiment 1. Columns correspond to users and rows to classifiers. Each cell $M(i, j)$ contains the number of posts (out of 20) for which classifier C_i answered *yes* for a post actually authored by user u_j . The diagonal is highlighted in boldface.

	352792	20307	117723	43315	161133	282143	353596	352809	13585	146319
C_{352792}	19	0	0	1	1	0	0	0	0	1
C_{20307}	7	17	6	5	9	9	2	3	2	9
C_{117723}	5	5	19	6	4	3	4	6	1	9
C_{43315}	9	2	1	17	2	0	2	6	1	8
C_{161133}	5	4	4	7	17	2	5	0	0	13
C_{282143}	4	9	9	8	2	14	4	4	0	9
C_{353596}	12	7	11	16	8	3	17	8	6	14
C_{352809}	3	6	4	8	4	3	8	10	0	9
C_{13585}	1	2	9	4	4	7	9	2	17	9
C_{146319}	12	5	7	10	10	4	9	5	1	17

- In order to evaluate the performance of each classifier, we make use of the following notions:
 - *True positives (tp)*: A test post written by user u_i is classified correctly by classifier C_i ; the number of true positives for C_i is found at position $M(i, i)$.
 - *False positives (fp)*: A test post written by user u_i is classified incorrectly by a classifier $C_j \neq C_i$; the number of false positives for C_i can be found by calculating $\sum_{j \neq i} M(i, j)$.
 - *True negatives (tn)*: A test post written by user u_i is classified correctly by a classifier $C_j \neq C_i$; the number of true negatives for each classifier is simply $(10 - 1) \cdot 20 - \sum_{i \neq j} M(i, j) = 180 - \sum_{i \neq j} M(i, j)$.
 - *False negatives (fn)*: A test post written by user u_i is classified incorrectly by classifier C_i ; the number of false negatives for C_i is calculated as $20 - M(i, i)$.

Based on the confusion matrix M and the above calculations, we can then derive:

$$precision(C_i) = \frac{tp(C_i)}{tp(C_i) + fp(C_i)}$$

and

$$recall(C_i) = \frac{tp(C_i)}{tp(C_i) + fn(C_i)}$$

which are standard metrics used to evaluate classifier performance. Finally, the harmonic mean of these two values, known as the F1 measure, is typically used as a good way to compare the performance of a set of classifiers.

Finally, we take the average of the three runs to obtain the final results, which we report next.

3.2. Results for Phase 1

The results of this set of experiments are shown in Figure 3 (classification performance) and Figure 4 (running time); each graph in the former shows the values obtained for precision, recall, and F1 measure. The typical tradeoff between precision and recall can be observed in each graph—though all instances had quite high values for recall, the classifiers that did best with respect to this measure did so at a high cost in precision (cf. MNB2, which boasted a recall of 0.96 but only 0.2 in precision. Figure 5 groups together the two best performers for each type of classifier.

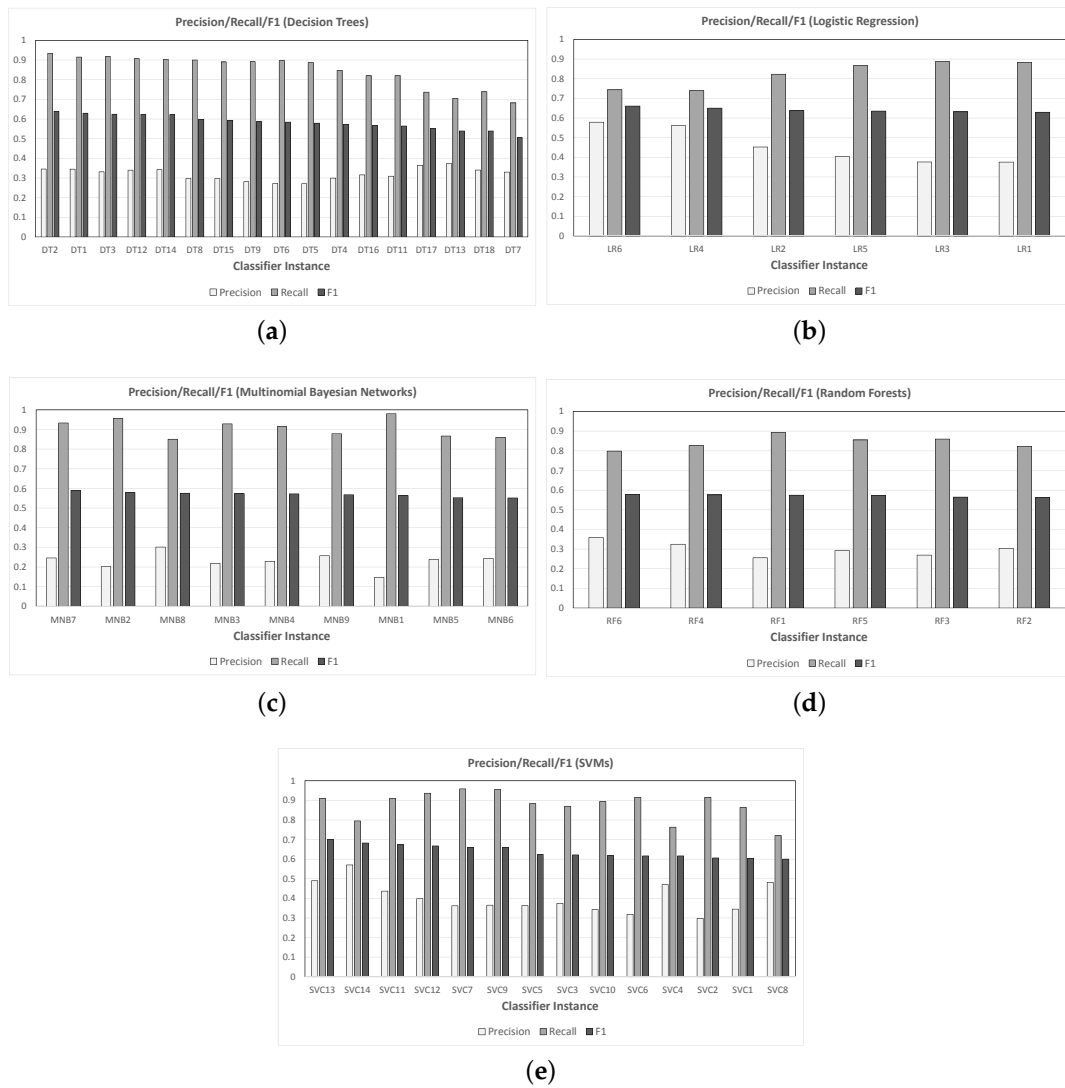


Figure 3. Precision, recall, and F1 values for all classifiers evaluated in Phase 1: (a) Decision Trees, (b) Logistic Regression, (c) Multinomial Bayesian Networks, (d) Random Forests, and (e) Support Vector Machines. In each graph, classifier instances are sorted in descending order of F1 measure.

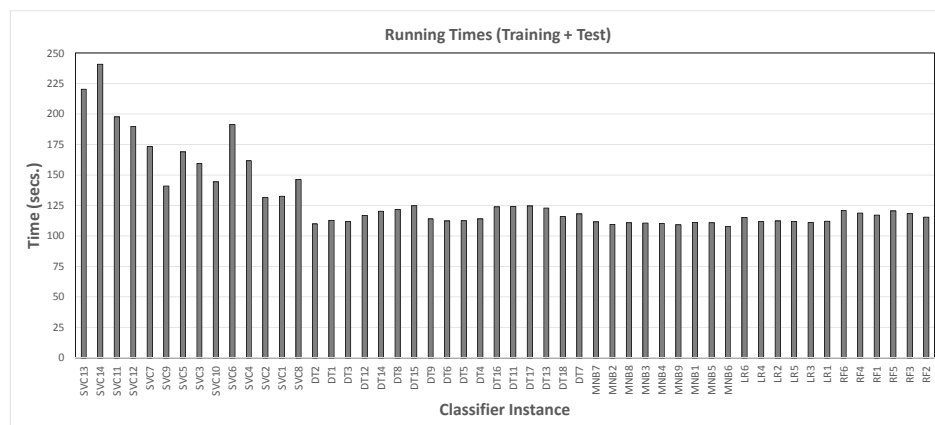


Figure 4. Running time taken to train and test each classifier instance from Phase 1, sorted by their F1 measure in the charts from Figure 3.

Note that, since there were 10 users in this phase, trying to solve the problem by random chance would succeed with probability 0.1. Therefore, the classifiers with the best precisions in Figure 5—SVC13, SVC14, LR6, and LR4—were 4.9 to 5.8 times better than this baseline.

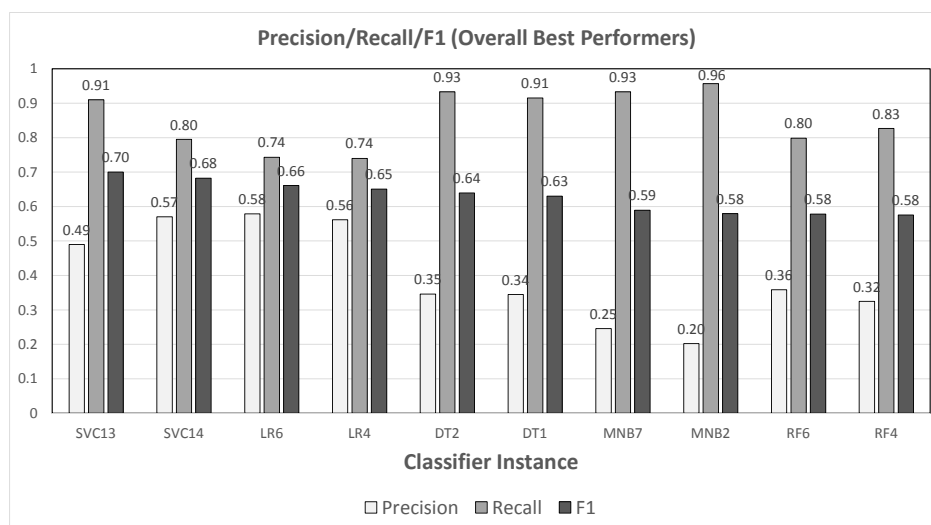


Figure 5. Best two performers from each chart in Figure 3, sorted by F1 measure.

3.3. Empirical Evaluation Phase 2: Seeding Known Duplicates

In this second part, we selected several classifiers—depending on their performance in Phase 1—and evaluated their capability to find duplicates. We took the best performers on their own, and also built several ensembles applying the *bootstrap aggregation* method, in which a set of classifiers is used in conjunction and a majority vote yields the final result. The choice was made among those shown in Figure 5, mostly by their performance with respect to F1 measure, which would yield SVC13 and SVC14. However, even though SVC14 performed slightly better than LR6, we chose the latter for the second pure classifier in order to favor diversity. The list of classifiers was the following:

- SVC13
- LR6
- E1 = {SVC13, SVC14, LR6, LR4}
- E2 = {SVC13, SVC14, LR6}
- E3 = {SVC13, SVC14, LR4}
- E4 = {SVC13, LR4, LR6}
- E5 = {SVC14, LR4, LR6}

Since, as discussed earlier, in this domain it is very difficult (or impossible) to obtain ground truth data, we engineered pairs of duplicates by simply dividing k users' posts into two sets and training independent classifiers with this data. Now, the definition of true/false positive and true/false negative depends on the type of pair being considered:

- For a pair of users that is known (or, rather, assumed) to be different:
 - *True negative*: classifier says no
 - *False positive*: classifier says yes

True positives and false negatives are impossible in this case.

- For a pair of users that is known to correspond to a duplicate:

- True positive: classifier says yes
- False negative: classifier says no

True negatives and false positives are impossible in this case.

Now, consider the task of choosing p pairs of users (u_i, u_j) at random and presenting 20 test posts for each user to the classifier corresponding to the opposite user (so, u_i 's posts to C_j , and vice versa). For $k = 2$, given that we have $54 + 2 = 56$ users (the original plus the duplicates), we have

$$\binom{54 + 2 = 56}{2} = 1540$$

possible pairs. In order to simulate a reasonable percentage of duplicates in the number of pairs tested, we conducted runs varying $p \in \{20, 40, 60, 80\}$ and calculated the resulting precision, recall, and F1 values, averaging over two runs of each—the results are presented in Figure 6.

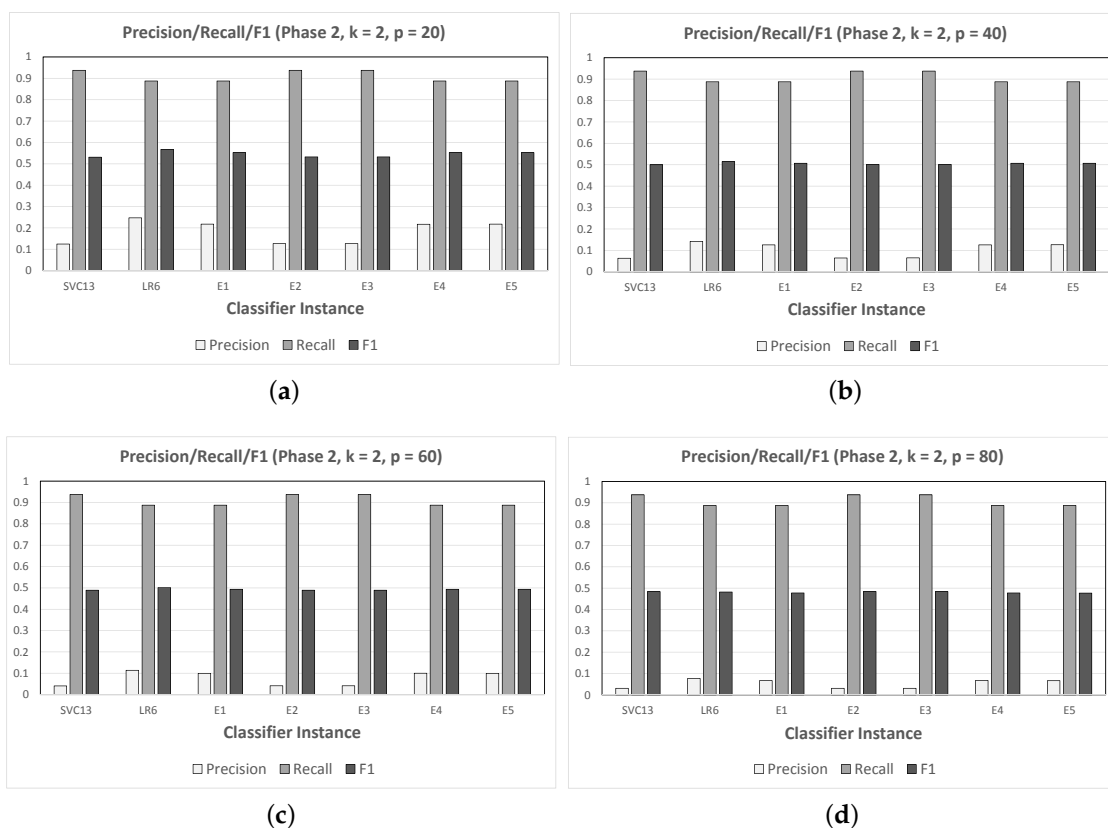


Figure 6. Evaluation of classification performance for chosen classifiers (two pure, five ensemble) over 54 users plus 2 duplicates engineered by splitting two of the original users, varying the number of pairs p sampled at random (guaranteed to contain the duplicate pairs): (a) $p = 20$, (b) $p = 40$, (c) $p = 60$, and (d) $p = 80$.

3.4. Results for Phase 2

The first thing that was evident when analyzing the results was that all classifiers' performance with respect to precision was much lower than those obtained in Phase 1. This can be explained by the sharp increase in the number of users with respect to the previous experiment (from 10 to 56)—giving many more opportunities for the classifiers to yield false positives. It should be noted, as before, that a classifier working by randomly answering *yes* one out of n times (for n users) would only have precision $1/56 \approx 0.0178$. Therefore, the values of 0.2468 (LR6) and 0.2176 (E1)—the two best

when $p = 20$ —were about 13.86 and 12.22 times higher, respectively. Furthermore, precision became lower as p increased, reaching 0.0769 (LR6) and 0.0673 (LR6) for $p = 80$ (still about 4.32 and 3.78 times better than chance, respectively). On the other hand, recall remained quite good for all classifiers, irrespective of the value of p , and was consistently better for SVC13—and all ensembles dominated by SVC classifiers—compared with LR6 (which had higher precision).

Finally, let us consider the last two steps of the workflow presented in Section 2, which involve the generation of *deduplication hypotheses*. Recall that what we are calling false positives in these experiments can actually be manifestations of unknown duplicates—pairs of user names that actually correspond to the same person. In order to deal with the relatively high incidence of such false positives that we saw above when we computed precision, we could set a threshold value t consisting of the number of times that posts presented to a different classifier trigger a positive response before we issue a deduplication hypothesis. For the same setting used in this experiment, we computed the number of deduplication hypotheses generated by each classifier (recall that the total number of unordered pairs was 1540 in this case, the percentage of this total is indicated in each case). So, for instance, SVC13 identified 94.28% of the pairs as potential duplicates, for threshold $t = 10$:

- For $t = 10$:
 - SVC13: 1452 (94.28%)
 - LR6: 451 (29.28%)
 - E1: 605 (39.28%)
 - E2: 1444 (93.76%)
 - E3: 1445 (93.83%)
 - E4: 609 (39.54%)
 - E5: 599 (38.89%)
- For $t = 15$:
 - SVC13: 1425 (92.53%)
 - LR6: 140 (9.09%)
 - E1: 204 (13.24%)
 - E2: 1396 (90.64%)
 - E3: 1396 (90.64%)
 - E4: 203 (13.18%)
 - E5: 207 (13.44%)

Two conclusions can be drawn from these results: First, the lower precision yielded by SVC13 in Phase 1 (as well as Phase 2 for the ensembles dominated by SVC-based classifiers) actually had farther-reaching causes than one might initially suspect by looking at the final quality measures, since the false positive rate was clearly quite high (as evidenced by the small change in number of hypotheses when increasing the value of t). Second, even though the performance of LR6 was also low in Phase 2, it was nonetheless capable of reducing the number of pairs to inspect by over 70% for $t = 10$ (i.e., at least half of the test posts), and over 90% for $t = 15$ (i.e., at least three-quarters of the test posts). This last phenomenon was also observable for the ensembles in which logistic regression-based classifiers had an important role (E1, E4, and E5).

Example 3. *The following posts are a few examples from a pair of users that, according to LR6, could have been written by the same person:*

- *Posts by user 353596:*

"lol i have that pasted to the front door of my office but if we really want to get technical here...these are all "cheap" translations of binary. i will type up a full explanation from home tonight. i also dug up and old piece of software that i have that is wonderful for quick lookups. i will post that as well."

"well "%path%" is the variable name. "path" is a term. it is always more important to learn terms and concepts rather than syntax that is specific to an environment. but yeah"

"didn't mean to burst your bubble about the script. i also saw some of the samples but none of them convinced me that the user thought they were really talking to a human being. most were prolly playing along like i do there are some msn ones also... anyone has visited they didn't sign up for the forums. but that's ok they can lurk for a while :whatsup:"

- Posts by user 352820:

"there are a few newish tools for but the issues with bt in the first place was poor implementation on behalf of the manufacturers, most of which were fixed."

"well if talking about power friendly then a hd in lan enclosure would be best. you could build something that would consume less power but it would be more expensive, unless you are looking for and upwards."

"the only good way to work in computer security straight is to be a "black hat" then at some point (arrested a few to many times, you get a wife/family) you decide you can't do it any more. the other way is to go into sysadmin and at some point, once you have experience going into nothing but security, u", i see way to many people about trying to sell themselves as pen-testers and security consultants and know fuck all, don't be one of those people."

Having identified this pair of users as a deduplication hypothesis, human expert analysts can then take a closer look into their posts, activities, and other data in order to confirm or reject it.

4. Related Work

Though there have been several approaches in the general areas of databases and cyber security/security informatics that tackle problems similar to adversarial deduplication, there are important differences. The following discusses the body of work in these two areas that is closest to our efforts.

Most of the research carried out towards the development of general tools for solving deduplication/entity resolution problems has been carried out in the databases community. Traditional approaches involve leveraging pairwise similarity over entity attributes [7], but other promising proposals are based on so-called *collective entity resolution* [8], which exploits additional relational information in the data since references to different entities may co-occur. An example of such an approach is called *iterative blocking* [9], which is based on iteratively grouping together matching records in blocks. This allows the use of the information processed so far to inform further decisions. Other approaches use similar techniques to the ones adopted here, in which machine learning classifiers are learned from examples and later used to determine whether or not an arbitrary pair of records are duplicates of each other based on a wide variety of features [8,10]. Recently, [11] defined a declarative framework for entity resolution based on *matching dependencies* (MDs). This formalism was first introduced in [12,13], and consists of declarative rules that generalize entity resolution tasks, eliminating duplicates via a matching process. These rules state that certain attribute values in relational tuples, under certain similarity conditions over possibly other attribute values in those tuples, must be made the same. The original semantics for MDs, defined in [13], was redefined and extended in [14].

As mentioned in the Introduction, most of the approaches developed in the databases literature operate under the assumption that duplications are the effect of clerical errors in data entry, inherent ambiguity in names or other attributes, inconsistent abbreviations and formatting, etc., rather than the fact that the objects (users in this case) are explicitly trying to obfuscate their identities [8–10]. As a result of this, and due to the nature of the information that can be collected from the type of source

we are looking at (forums and other sites in the Dark Web), the dataset does not contain the relational information and structure needed for these proposals to be applied.

In the security informatics/cybersecurity literature, the general problem of identifying deceptive behavior with respect to identity has been around for almost two decades—to the best of our knowledge, [15] is the earliest appearance of this issue, where the issue of anonymity is central. In a similar vein, [16,17] are other works on detecting aliases, also with a focus on anonymity guarantees. Historically, stylometry is one of the more widely used methods for authorship analysis. It studies the personal characteristics of individuals' writing style—a survey of recent approaches to the authorship problem can be found in [18], and some notable works include [19–21]. In [22], the authors apply stylometry techniques towards attributing authorship of instant messages, making use of features such as frequency distributions for characters, words, emoticons, function words, short words, abbreviations, and punctuation, average word length, and average words per sentence, whether or not the message contains a greeting/farewell, and spelling/grammatical errors. It is possible that extending our approach with a deeper analysis of features such as these for specific forums and marketplaces would help us improve the accuracy of our methods. One of the problems with the application of such technique(s) to the kind of data we seek to analyze is the length of the texts, which are considerably short and therefore identification (classification) accuracy suffers. Not surprisingly, even just a few short sentences can carry a great deal of information that a human analyst can use for identification, but that information goes beyond the style of the writing. Additional techniques that are capable of exploiting specific domain information are needed. The work of [23] suggests that contextual analysis and tolerance to uncertainty are especially needed for the identification of individuals in social media forensics. Tsikerdekis and Zeadally [24], on the other hand, also incorporate non-verbal behavior to aid user identification.

Another work that is closely related to ours is that of [25]. In that paper, the authors focus on the related problem of *authorship matching*, which seeks to validate whether or not two accounts having the *same username* on multiple Dark Web forums belong to the same person through writing style analysis (stylometry) and SVMs. An N two-way classification model, where N is the number of authors being tested, was trained with a set of ten active users (the ones with at least 400 posts and around 6000 words), divided into two parts, each containing half the posts of each user. In a validation phase, the best parameters for the model were found, in order to test it against the forum post of another set of accounts in a different Dark Web forum having the same username as the ones used in the validation phase. Using different types of similarity functions to evaluate the performance, their approach yielded around 80% accuracy. The authorship matching problem is based on the assumption that there is a tendency in users to adopt similar usernames for their accounts in different sites. Though the results are promising, this assumption is quite strong and is not always valid in adversarial settings where users try to obfuscate their identities. In such scenarios, looking for equal or syntactically similar user names does not seem to be reasonable.

The problem of detecting *sock puppets* is also quite related to the general problem of reasoning about author identities [26–28]. The term “sock puppet” is commonly applied to multiple user accounts that are created and used with the purpose of creating the illusion of support within a community, such as artificially increasing the number of positive votes on a social media post. Though some of the underlying techniques applied to this problem are shared with the other work discussed here, there is one important distinguishing characteristic: sock puppets inherently involve interactions among themselves and the original user, whereas in adversarial deduplication settings it is probably the case that different profiles created by the same person do not interact.

The work that is perhaps the closest in spirit to the results that we show here is [29], in which the authors tackle the problems of determining aliases and attributing authorship using machine learning classifiers that work with three kinds of features: character-level n -grams, stylometry, and timestamps. Though their approach to determining aliases shares several aspects with our own, the main differences lie in that their solutions involve a high degree of specialization, since single

topic posts are used in the evaluation, stylometry-based techniques that depend on the specific data, and analysis of timestamps. On the other other hand, our goal is to develop a general approach that (i) requires minimal domain-specific preparation; (ii) is flexible with respect to the set of users in the domain (i.e., it should be easy to add or remove users); (iii) makes as few assumptions as possible regarding user behavior.

Finally, the same kind of analysis based on applying machine learning techniques such as classifiers and clustering algorithms has been successfully adopted in other problems related to cyber security, such as the identification of product offerings in malicious hacker markets [30], at-risk system identification [31], and exploit prediction [32]. Though these are difficult problems, their advantage over adversarial deduplication is the availability of some kind of ground truth that can be used to evaluate and tune the proposed solutions.

5. Conclusions and Future Work

In this paper we tackle the so-called *adversarial deduplication* problem, which seeks to identify pairs of users who are actively trying to hide their identity by creating multiple profiles. We argue that this problem is fundamentally different from but closely related to the traditional entity resolution or deduplication problem in databases, since this problem is assumed to arise as a consequence of unintentional errors. We focus on the cyber-security setting of malicious hacker forums and marketplaces on the dark web, where such intentional obfuscation is the norm.

As a first step towards developing tools to address this problem, we proposed the use of machine learning classifiers trained to identify text-based features, and designed a set of experiments to evaluate their effectiveness. Our preliminary results are promising in that reasonably high precision and recall could be obtained in an initial training and evaluation phase with few users. In a second phase with over 50 users, the precision became much lower due to the incidence of false positives. However, since the overall goal of the approach is to create deduplication hypotheses that are then passed on to human analysts for further review, an additional threshold parameter could be applied to manage the number of generated hypotheses. Furthermore, it should be noted that in a system deployed in the real world, classifiers may need to be periodically retrained in order to incorporate information from new posts, as well as new classifiers trained for new users. The specific period for carrying out these operations will depend on the details of the domain in question.

Future work includes carrying out further experiments with other datasets, and identifying or learning other features (potentially quite complex) to incorporate them to the task, aiming to capture the context in which these posts are issued, such as topics in which users post, co-occurrence with other users, abnormality in learned behaviors, etc.

Author Contributions: Conceptualization, J.N.P., G.I.S., M.V.M., and M.A.F.; Data curation, J.N.P.; Formal analysis, G.I.S. and M.V.M.; Funding acquisition, M.A.F.; Methodology, J.N.P., G.I.S., and M.V.M.; Project administration, M.A.F.; Resources, M.A.F.; Software, J.N.P.; Supervision, G.I.S. and M.A.F.; Validation, G.I.S. and M.V.M.; Writing—original draft, J.N.P., G.I.S., and M.V.M.; Writing—review and editing, G.I.S., M.V.M., and M.A.F.

Funding: This research was funded by Universidad Nacional del Sur (UNS) and CONICET, Argentina, by the U.S. Department of the Navy, Office of Naval Research, grant N00014-15-1-2742, and by the EU H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 690974 for the project “MIREL”.

Acknowledgments: We are grateful to CYR3CON (<https://cyr3con.ai>) for providing access to the dataset used in our experiments. Images in Figure 1 designed by Freepik from Flaticon (<http://www.freepik.com>)—used with permission.

Conflicts of Interest: The funding sponsors had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, and in the decision to publish the results.

References

1. Elmagarmid, A.K.; Ipeirotis, P.G.; Verykios, V.S. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* **2007**, *19*, 1–16. [[CrossRef](#)]
2. Bleiholder, J.; Naumann, F. Data Fusion. *ACM Comput. Surv.* **2009**, *41*, 1–41. [[CrossRef](#)]

3. Nunes, E.; Diab, A.; Gunn, A.T.; Marin, E.; Mishra, V.; Paliath, V.; Robertson, J.; Shakarian, J.; Thart, A.; Shakarian, P. Darknet and deepnet mining for proactive cybersecurity threat intelligence. *arXiv* **2016**, arXiv:1607.08583.
4. NIST. National Vulnerability Database. Available online: <https://nvd.nist.gov/> (accessed on 24 July 2018).
5. CVE. Common Vulnerabilities and Exposures: The Standard for Information Security Vulnerability Names. Available online: <http://cve.mitre.org/> (accessed on 24 July 2018).
6. Shakarian, J.; Gunn, A.T.; Shakarian, P. Exploring Malicious Hacker Forums. In *Cyber Deception, Building the Scientific Foundation*; Springer: Cham, Switzerland, 2016; pp. 261–284.
7. Getoor, L.; Machanavajjhala, A. Entity Resolution: Theory, Practice and Open Challenges. *Proc. VLDB Endow.* **2012**, *5*, 2018–2019. [[CrossRef](#)]
8. Bhattacharya, I.; Getoor, L. Collective Entity Resolution in Relational Data. *ACM Trans. Knowl. Discov. Data* **2007**, *1*, 5. [[CrossRef](#)]
9. Whang, S.E.; Menestrina, D.; Koutrika, G.; Theobald, M.; Garcia-Molina, H. Entity Resolution with Iterative Blocking. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, Providence, RI, USA, 29 June–2 July 2009.
10. Bhattacharya, I.; Getoor, L. Query-time entity resolution. *J. Artif. Intell. Res.* **2007**, *30*, 621–657. [[CrossRef](#)]
11. Bahmani, Z.; Bertossi, L.E.; Vasiloglou, N. ERBlox: Combining matching dependencies with machine learning for entity resolution. *Int. J. Approx. Reason.* **2017**, *83*, 118–141. [[CrossRef](#)]
12. Fan, W. Dependencies Revisited for Improving Data Quality. In Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Vancouver, BC, Canada, 9–12 June 2008; pp. 159–170.
13. Fan, W.; Jia, X.; Li, J.; Ma, S. Reasoning About Record Matching Rules. *Proc. VLDB Endow.* **2009**, *2*, 407–418. [[CrossRef](#)]
14. Bertossi, L.E.; Kolahi, S.; Lakshmanan, L.V.S. Data Cleaning and Query Answering with Matching Dependencies and Matching Functions. *Theory Comput. Syst.* **2013**, *52*, 441–482. [[CrossRef](#)]
15. Rao, J.R.; Rohatgi, P. Can pseudonymity really guarantee privacy? In Proceedings of the 9th USENIX Security Symposium, Denver, CO, USA, 14–17 August 2000; pp. 85–96.
16. Novak, J.; Raghavan, P.; Tomkins, A. Anti-aliasing on the web. In Proceedings of the 13th International Conference on World Wide Web, Manhattan, NY, USA, 17–22 May 2004; pp. 30–39.
17. Brennan, M.; Afroz, S.; Greenstadt, R. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Trans. Inf. Syst. Secur.* **2012**, *15*, 12. [[CrossRef](#)]
18. Swain, S.; Mishra, G.; Sindhu, C. Recent approaches on authorship attribution techniques: An overview. In Proceedings of the 2017 International Conference of Electronics, Communication and Aerospace Technology, Tamil Nadu, India, 20–22 April 2017; pp. 557–566.
19. Abbasi, A.; Chen, H. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.* **2008**, *26*, 7. [[CrossRef](#)]
20. Narayanan, A.; Paskov, H.; Gong, N.Z.; Bethencourt, J.; Stefanov, E.; Shin, E.C.R.; Song, D. On the feasibility of internet-scale author identification. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 300–314.
21. Johansson, F.; Kaati, L.; Shrestha, A. Detecting multiple aliases in social media. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Niagara Falls, ON, Canada, 25–28 August 2013; pp. 1004–1011.
22. Orebaugh, A.; Allnutt, J. Classification of instant messaging communications for forensics analysis. *Int. J. Forensic Comput. Sci.* **2009**, *1*, 22–28. [[CrossRef](#)]
23. Rocha, A.; Scheirer, W.J.; Forstall, C.W.; Cavalcante, T.; Theophilo, A.; Shen, B.; Carvalho, A.R.B.; Stamatatos, E. Authorship Attribution for Social Media Forensics. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 5–33. [[CrossRef](#)]
24. Tsikerdekis, M.; Zeadally, S. Multiple account identity deception detection in social media using nonverbal behavior. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 1311–1321. [[CrossRef](#)]
25. Ho, T.N.; Ng, W.K. Application of Stylometry to DarkWeb Forum User Identification. In Proceedings of the International Conference on Information and Communications Security, Singapore, 29 November–2 December 2016.

26. Zheng, X.; Lai, Y.M.; Chow, K.P.; Hui, L.C.; Yiu, S.M. Sockpuppet detection in online discussion forums. In Proceedings of the Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Dalian, China, 14–16 October 2011; pp. 374–377.
27. Kumar, S.; Cheng, J.; Leskovec, J.; Subrahmanian, V. An army of me: Sockpuppets in online discussion communities. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 857–866.
28. Yamak, Z.; Saunier, J.; Vercouter, L. SocksCatch: Automatic detection and grouping of sockpuppets in social media. *Knowl.-Based Syst.* **2018**, *149*, 124–142. [[CrossRef](#)]
29. Spitters, M.; Klaver, F.; Koot, G.; van Staalduinen, M. Authorship analysis on dark marketplace forums. In Proceedings of the European Intelligence and Security Informatics Conference, Manchester, UK, 7–9 September 2015; pp. 1–8.
30. Marin, E.; Diab, A.; Shakarian, P. Product offerings in malicious hacker markets. In Proceedings of the IEEE Intelligence and Security Informatics 2016 Conference, Tucson, Arizona, USA, 27–30 September 2016; pp. 187–189.
31. Nunes, E.; Shakarian, P.; Simari, G.I. At-risk system identification via analysis of discussions on the darkweb. In Proceedings of the APWG Symposium on Electronic Crime Research, San Diego, CA, USA, 15–17 May 2018; pp. 1–12.
32. Tavabi, N.; Goyal, P.; Almukaynizi, M.; Shakarian, P.; Lerman, K. DarkEmbed: Exploit Prediction with Neural Language Models. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, LA, USA, 2–7 February 2018.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).