COMP530 - Final Report
Profile Matching

Erhan Tezcan · Waris Gill · Mandana Bagheri Marzijarani
etezcan19@ku.edu.tr · wgill18@ku.edu.tr · mmarzijarani20@ku.edu.tr

18.01.2021

1 Abstract

The emergence of Online Social Networks (OSNs) and their increasing role in our daily lives have created a lot of opportunity for research. Most importantly, OSNs are a source of privacy concerns, not just with regards to how user data is handled on a particular platform, more-so with regards to the aggregation of user information across multiple OSNs. From the perspective of online service providers, this aggregate information is a goldmine in understanding their customers and their behaviors, which is a major cause for concern from the privacy point of view.

In this work, we have implemented different methods for matching user profiles across two popular OSNs (Facebook and Twitter) using multiple state-of-the-art machine learning models and algorithms. These methods enabled us to cross-match users via their publicly available information on each platform. We have evaluated our implementation on a dataset consisting of about 1000 Facebook and 1000 Twitter profiles. We believe that this work –although preliminary– is a step in the right direction in demonstrating privacy risks of users sharing habits on online platforms.

2 Introduction

Over the past decade numerous Online Social Networks (OSN), such as Facebook and Twitter have emerged. Based on the services and features that they offer, each attracts different demographics as users. The recent statistics show that the average number of OSN accounts per person is 8.6 [15]. The commonality of multiple accounts per user has created significant opportunity for profile matching research. Such research intends to match users with the same identity across multiple OSNs using a wide array of techniques, such as state-of-the-art machine learning models [8]. Profile matching can be used in aggregating user information among different OSNs and in providing an insight on each user unachievable over a single platform. That is the reason why builders of online services have a keen interest in matching user profile across different OSNs, i.e., so that they can create better recommendation systems and consequently provide more accuracy in advertisement.

From a security and privacy standpoint, effective profile matching raises serious concerns. Publicly available information of users across different platforms can be used to their disadvantage if an attacker is able to link profiles of individuals on one platform to their profile in another. Such privacy attacks can reveal sensitive information about individuals against their wish, and can ultimately lead to real world dangers. For example, the identity of a user can be obtained from one platform and her address can be obtained from a matched profile on another platform. These information can be used to threaten the user in real world. The importance of profile matching research lies in the fact that it can quantify and ultimately mitigate the risk of such privacy attacks and help OSNs harden their policies to safeguard user privacy.

In this research we have aimed at building a Machine Learning (ML) model for profile matching that improves over previous efforts to quantify and expose the risks of such privacy attacks by using Facebook and Twitter public profile information. Our proposed methods matches user profiles across Facebook and Twitter accounts by using machine learning and direct/indirect matching techniques. We evaluated our work on a dataset composed of 2000 Facebook and Twitter accounts' publicly available data. in profile matching. The main contributions of this work can be summarized as follows:

- Collecting and organizing a dataset of about 2000 Facebook/Twitter users public profiles.
- Up to date Facebook and Twitter scrapping tools that can collect user public profile on demand.
- Direct and indirect user profile matching methods.
- A machine learning based profile matching model.
- Study of effects of different publicly available attributes on success of profile matching
- Evaluation of our methods on the collected dataset.

2.1 Problem Statement

Given the variety of social networks, the users create accounts in different ways as the rules and regulations are different from one another. However, almost all of these have a few things in common: i.e. every user has a username, some content, and friends as well as followers. The attributes like username, content, and followers will vary from one social media platform to another. So it is quite challenging to match a single user's profiles among many OSN's, as there is no unique global identifier (e.g.: username) for the user. This problem is defined as: "Identity Resolution in Online Social Networks (IROSNs)". IROSNs can be defined as: "Given the user attributes (e.g.: name, username, location, content etc.) in a social networking site, identify him/her on another social network platform". For instance, given a user's Facebook profile, find his/her Twitter profile. IROSNs techniques has the following applications:

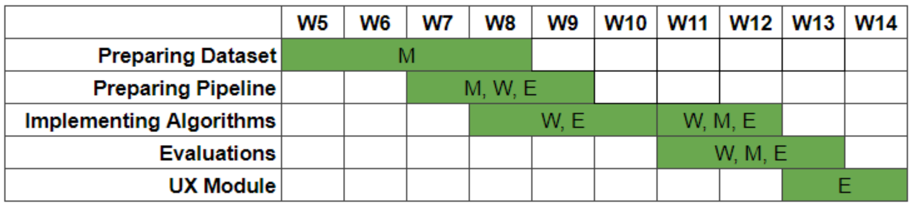
1. **Protecting User Privacy:** IROSNs tools can protect/increase the privacy of a user. On some social media sites, the users hide their personal information and just share their personal views (about religion, politics, etc.) while on some they have some public information such as username, region, post time, followers who may have similar views, but they are posting them publicly. So, in such cases, an attacker can easily learn who

is the anonymized user which is very critical and dangerous in some scenarios. Thus, it is very important to develop a privacy tool which can quantized the anonymity level of user profile and suggest ways to increase its privacy.

2. **Identifying Malicious Users:** With the help of IROSNs, malicious users can be identified on multiple platforms. The malicious users obfuscate their attributes so that they cannot be detected on other platforms. So, behavioural-based IROSNs can be utilized to find and discard such users on online social networks.

3. **Recommendation Systems:** IROSNs can also suggest friends based on the information of one network on another. Additionally, it can also suggest content such as music, videos, and articles based on the user's interests and posts.

2.2 Project Timeline



	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Preparing Dataset	🟩 M	🟩	🟩	🟩						
Preparing Pipeline			🟩 M, W, E	🟩	🟩					
Implementing Algorithms				🟩 W, E	🟩	🟩	🟩 W, M, E	🟩		
Evaluations							🟩 W, M, E	🟩	🟩	
UX Module									🟩 E	🟩

Figure 1: GANTT Chart.

In figure 1 we present our GANTT chart for this project, as revised in the progress report. First, we collected the dataset, which took a lot of time due to several changes in the methodology regarding us not being able to use the API. Then, we have implemented the direct matching algorithm, with a naive version. The similarity functions used in this direct matching algorithm were thoroughly researched, with respect to the existing works. The overall code structure was also being completed by the end of second task. Then, we have worked on using machine learning for direct matching, while also implementing the indirect matching algorithm. At the end, we have implemented the UX module.

The rest of the paper is organized as follows. In Section 3, we recall the related work and what makes our effort different from the former ones. In Section 4, We discuss the data collection and the challenges we face and how to overcame those challenges . In Section 5, we explain the details of our proposed methods. In Section 6 we evaluate our profile matching methods by using them on real world dataset. At last in Section 7, we will discuss limitation, future work and conclude this effort.

3 Related Work

In this section we review the related research that has been done in profile matching area. Mostly, the former endeavours focused on user's public attributes such as biography, name, location, self description etc. [3] [4] [6] or the user's behaviours such as Facebook posts and Twitter's tweets [11] [5], some work such as [7] use both public attributes and user behaviour in linking profiles. The other method used in recent research relies solely on image identification [12]. These methods each show promise as discussed in [9] [8] [12] these methods all rely on the data that is not mandatory and can be very noisy which makes their accuracy rate very unreliable. In this work, we use up-to-date dataset obtained from Facebook and Twitter Scrapper that we implemented and instead of focusing only on one method, we use a combination direct/indirect and state-of-the-art Natural Language Processing (NLP) models and Machine Learning classifiers such as AdaBoost and tried to improve over the existing methods of profile matching.

4 Data Collection

As far as we are aware, there is no publicly available dataset that contains Twitter and Facebook public profiles linked together. As such, we had to systematically collect the data of Twitter and Facebook users ourselves. As mentioned in the progress report, We had originally planned to create a dataset of 1000 Facebook and 1000 Twitter accounts, where 20% of the profiles composed our ground truth (i.e., manually matched). It is worthy to mention that this decision is motivated by the works of [6, 8, 13].

Before starting to collect the data, the list of Facebook and Twitter users was prepared manually, The data was divided into two categories: matched and unmatched. To build the list of 200 profiles across Twitter and Facebook that have the same identity (i.e., matched), we used a list of publicly known figures that have confirmed profiles on both platforms. However, we did not want our ground truth to be strongly biased towards famous figures. As such, we kept the number of public figures in our ground truth under 50, i.e., 25% of the matched ground truth. The rest of the ground truth includes regular people who were manually confirmed on both platforms using their image, friends/following list, affiliation, etc.,

For the rest of the dataset, we tried to minimize the possibility of matched profiles by selecting users on Facebook and Twitter who belong to two completely different social circles. For Twitter, we selected 800 people with cybersecurity, IT and academic backgrounds. For Facebook we started by selecting people who worked in an auto-dealership in Los Angeles and collected the usernames of their friends.

To collect the selected users' publicly available data, our initial plan (as described in the project proposal) was to use Twitter's developer API and Facebook's Graph API. It is worthwhile to mention that we took Dr. Emre hoca's advice seriously and investigated the practicality of our approach before proposing it. While we submitted the proposal, our application to open a Twitter developer account was under review and we had already submitted a simple Facebook App for review. We had also investigated the daily rate limit of both APIs and their workarounds. Unfortunately our request to create a Twitter developer account was rejected three times, despite trying with the university email account and explaining thoroughly and exactly what we meant to use the data for. The most likely reason for rejections are concerns for US elections on these platforms and the risk of automated activity associated with such APIs.

As an alternative, we decided to look for publicly available online Facebook/Twitter data scrappers. There are many projects available on GitHub and we tried the top 5 among them. To the best of our knowledge there were no Facebook/Twitter scrapper that were both functional and up-to-date. This is most likely because both platforms regularly change their page structures and technologies and the crawlers are usually outdated. Consequently, we developed a Facebook and Twitter scrapper using Python and Selenium [18], coupled with jQuery selectors. The scrapper opens up Twitter and Facebook, logs into them, then iterates over desired profiles and several pages of each profile. On each page, the scrapper injects jQuery into the page, then uses jQuery selectors and helpers to select the desired data and return it as a dictionary. These dictionaries are then combined and saved as JSON files.

Several of such pages required scrolling down dozens of times and waiting for more data to be loaded (e.g., friends list on Facebook and Twitter). Furthermore, a particularly challenging issue was the API rate limits available on both platforms. For a typical profile, we would need to send 20 to 100 requests to the platform. Not only is sending these many requests slow, it also typically results in the account and/or the IP address being rate limited and further access denied for an unspecified amount of time.

The entirety of each profile's information was collected in two steps. In the first step, we collected the general information of the profile on both platforms, sans the friends list for Facebook and followers/followings list for Twitter. For this first step, we were able to work around platforms' rate limiting using VPNs and changing IPs every 20-40 profiles, before the platform would block our IP for unusual activities.

In step two, for a given profile we had to first load the friend, follower and following list pages, then scroll down multiple times (each scroll would load about 5 to 20 new friends), and then extract the usernames of the friends list. In the case of Facebook, we were able to collect the friend list using VPNs, as Facebook did not flag the account we used for logging in and collecting the data (only the IP address was flagged, and a sufficiently diverse VPN was enough of a workaround for the rate limit). Even though the process was extraordinarily slow, we were able to collect the list of the first 100 friends of about 1000 Facebook profiles.

In the case of Twitter, contrary to profile information, to collect the follower/following list, authentication with a valid account was required. Furthermore, Twitter employs several mechanisms to prevent logging in automatically. As such, the login step had to be done manually each time the scrapper was launched. Twitter repeatedly suspended the accounts we used for logging in. To circumvent this limitation, every time we used a new VPN connection and a new account, which would get suspended after scrapping a few additional profiles.

Another issue with Twitter was that after loading a follower list and scrolling down to get the entirety of the list loaded, the items higher on the list would be removed from the page and disappear. Given the aforementioned challenges with Twitter, the only data that we were **not** able to automatically collect successfully was the follower/following list. Given the fact that we already had Facebook friend's list, we had to come up with a way to collect Twitter's follower list so each of our team members manually collected the followers list of each Twitter profile from a Twitter app called Vicinitas [17] and we were able to import the Twitter's follower list to our dataset.

After collecting about 1950 profile's information across Facebook and Twitter including almost 20 percent matched profiles, we imported the data into a MongoDB database hosted on a Vultr [16] cloud server. Our database has two collections, Facebook and Twitter, each of which is indexed by username as id.

5 Proposed Models

The heart of this project is our Direct Matching algorithm. We will first describe it, and then we will describe our similarity algorithms. Then, we will discuss the weight problem, and how we have used machine learning to solve it. We will also describe indirect matching.

5.1 Direct Matching

Direct matching algorithm compares a user in the source network to all users in the target network. Then, it chooses the most similar user as a possible candidate. Let N_s be the source network and N_t be the target network. Also denote F as the feature set, and W as the weights, where W_f denotes the weight for some feature f. For some source user $u \in N_s$, the match is found by the following:

$$t = \underset{c \in N_t}{\operatorname{argmax}} \sum_{f \in F} W_f \times S_f(u_f, c_f) \tag{1}$$

One thing to notice here is that equation 1 always returns a match, no matter how low the score is. However, the main problem here is the weights W. How do we decide on these? We have used Machine Learning to overcome this problem with great success. However, first we should talk about our feature set.

There are 6 features that are taken into account, each with their own similarity function S_f:

1. **Username** similarity is calculated using **Jaro distance**.
2. **Name** similarity is calculated using Jaro distance.
3. **Location** similarity is calculated using Jaro distance.
4. **Biography** similarity is calculated by first creating text embeddings with one of the state of the art NLP model **BERT**, and then measuring the **cosine distance**. For the Facebook users, the **Headline** feature is also concatenated to the biography, as it contains important information what could have been written in the biography, and the feature by itself doesn't have a counterpart on Twitter.
5. **Birthday** similarity is calculated as 1 if they are the same, 0 otherwise. This binary decision is made because we do not think different birth dates would be entered on separate accounts of the same user, and computing similarity based on dates would not make sense if they were different.
6. **Website** similarity is calculated using Jaro distance. Note that sometimes the person has a homepage, or blog, or for developers perhaps their GitHub profiles etc. and these can be found in both networks.

We have used Jaro distance especially, as it is very powerful for short text, and it respects word positions. For example, “MikeDuhn” and “DuhnMike” have very high similarity with Jaro distance, though letters are different with respect to their positions.

We have thought of using profile images as a feature too, however Facebook image URLs expired. This is also important, as it brings forth a shortcoming of storing the crawled data. We were storing URLs instead of images, but this shows that we should have actually downloaded the image.

Another feature that we were planning on to use was an artificially created one called NER, which stands for Named Entity Recognition. By using state-of-the-art NLP models, we were planning to extract named entities from biography, location, education, work etc. but we have later realized that biography itself is enough, and our text similarity algorithm, based on **BERT**, is actually doing the NER in itself.

5.1.1 Direct Matching with Machine Learning

In equation 1 there is the problem of deciding on weights. Thankfully, machine learning helps us exactly there, so we have trained a model to work with.

First, we retrieve the ground-truth data from our dataset. This ground-truth data is such that for every user in one domain we know the username of it’s profile on the other domain. For every user, we randomly sample non-matching users, 6 of them in our work, and then together with the matching we calculate similarity scores for each feature. After doing this for every user in the ground-truth, we obtain a dataset where the features are similarity scores, and the label is either 1 or 0, i.e matching or not-matching.

We have tried several models, and for each of them we performed a grid search to find the best hyper-parameters with **Stratified 5-Folds** cross-validation. In stratified the folds are formed by maintaining the percentage of samples for each class.

- AdaBoostClassifier *
- ExtraTreesClassifier
- RandomForestClassifier
- GradientBoostingClassifier
- BaggingClassifier

AdaBoostClassifier has shown the best performance, so we have used it in our direct matching algorithm. These models can also be trained and evaluated in a Python notebook, which is included in our submission as `ml.ipynb`. The resulting direct matching algorithm is as follows:

$$t = \underset{c \in N_t}{\operatorname{argmax}} \left[\texttt{Predict}(\underset{f \in F}{S_f} (u_f, c_f)) = 1 \right] \tag{2}$$

Here, `Predict` predicts whether there is a match or not by looking at the similarity scores using the trained AdaBoostClassifier. If we had just returned the resulting label, we would have to consider multiple matches after the predictions. Instead of this, we return the label probabilities, where if the probability of being a match is higher than not being a match, we can consider it a match. When there are multiple matches, we pick the highly probable one.

This shows two major differences compared to the naive direct matching:

1. The naive algorithm returned a target user no matter how low the score is. However, as seen in equation 2, if the model does not predict any matches, than we do not return anything.

2. We do not have a preset weight list, as they are part of the trained network, and they depend on the data.

5.2 Indirect Matching

The indirect matching algorithm works on top of the direct matching algorithm. This is more computationally demanding and should be used when direct matching does not yield confident results. We describe indirect matching in algorithm 1. Here, the function `DirectMatch` is the matching algorithm we defined in the

Algorithm 1 Indirect Matching

```
On input $u \in N_s$, first retrieve the friends/followers $M$ of user $u$.
for all $m \in M$ do
  Find direct match of $m$ as $m' = $ DirectMatch($m$). Call the resulting set $M'$.
end for
Find common friends of users in $M'$. This results in a key-value pair $(k, v)$ where $k$ is the username and $v$ is the number of users it is friends to in $M'$.
Pick the target user as $k$ that has the maximum $v$, i.e. the "most-common common friend".
```

previous subsection. The intuition behind indirect matching algorithm is that, if one user can't be directly matched but their friends can be, than perhaps the same is true on the target network. We also guess that these friends are connected on both networks.

5.3 UX Module

The last task of our project was to create a better user experience by outputting HTML file for a successful match, describing how the users are matched and how are the similarity scores. We have included several screenshots in Appendix B regarding this.

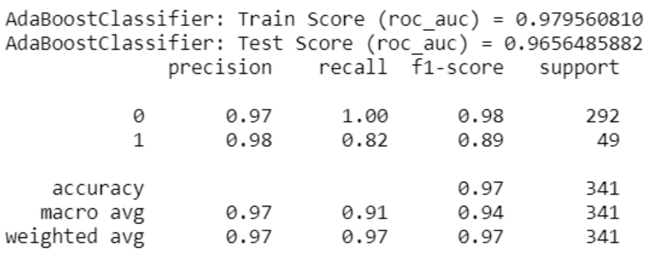
6 Evaluations

In this section, we will evaluate the performance of our ML based direct matching method. We are not evaluating the naive method and indirect matching, because the naive method is not to be used when we can use ML-based approach, and indirect matching requires a more complete dataset, and in our case we sometimes do not have all the users we need to search the network.

6.1 Performance

We have measured the performance of the model, by calculating the evaluation metrics on it. Note that we train our model based on the ground-truth, however our dataset is actually larger than what was used in the training model. Since it would not be feasible to run this code over all 1000×1000 user pairs across networks, we rely on the performance metrics for the machine learning model and how it performs against training and test data.

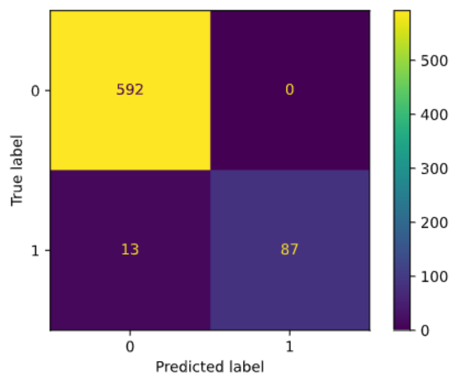
Our main performance metrics are Precision, Recall and Accuracy, as defined in Appendix A. Precision and Recall are especially important because we have seen that accuracy alone might not be enough, as mentioned in [2]. These metrics are shown in the Fig. 2. Additionally, the confusion matrix for the training and test is shown in the figures 3a and 3b, respectively.



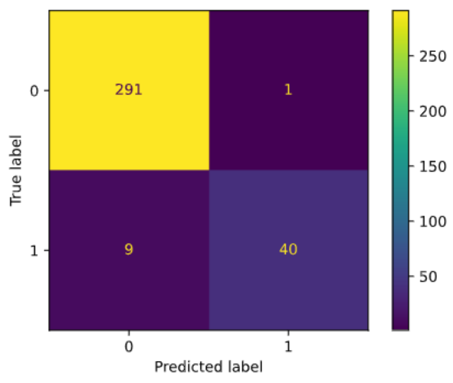
```
AdaBoostClassifier: Train Score (roc_auc) = 0.979560810
AdaBoostClassifier: Test Score (roc_auc) = 0.9656485882
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       292
           1       0.98      0.82      0.89        49

    accuracy                           0.97       341
   macro avg       0.97      0.91      0.94       341
weighted avg       0.97      0.97      0.97       341
```

Figure 2: AdaBoostClassifier classification report on the test data.

In figure 2 we see the performance of AdaBoostClassifier. It has an exceptionally high precision, recall and accuracy. We can confidently say that our model is working well for our set of features and similarity functions. Also notice that our `ROC AUC` scores for both training and test data are around ≈ 0.97 which is very good.



(a) Confusion Matrix On Train Dataset with 5-fold cross validation.



(b) Confusion Matrix On Test Dataset for the same model.

Figure 3: AdaBoostClassifier confusion matrices.

In figure 3 we can see the confusion matrices of AdaBoostClassifier on both the training and test data. We can see that the number of false positives and false negatives are very low. In particular, the model almost never classifies a non-matching pair as a match (when true label is 0 but is predicted 1). Rather, the model seems to miss several matches. We are confident that these problems can also be dealt with by improving our dataset. We were not particularly diligent about how much of the features exist for each pair in our training set, so there could be a bias towards some features which cause these errors. For example, “MikeDuhn” and “DuhnMike” may be the same person, which increases the weight of the username, but “Mike1” and “Mike2” may be different people, thus the weight may cause the model to trust this similarity more than it should.

6.2 Feasibility

The matching algorithms we have implemented are computationally demanding. The training itself is not that hard, but calculating the similarities are the part that takes a lot of time. The complexity is $\mathcal{O}(n)$ for n users in the target domain for direct matching. For indirect matching, the complexity becomes $\mathcal{O}(mn)$ for m friends of the source user and n users in the target domain. Each pair of users contribute greatly to the runtime, as their features are to be matched. Note that if the source user does not have some features (such as a user without location, biography, etc.) then of course the process ends faster, however then the model will only have the names to work with.

The argument here is that, though this process takes a lot of effort to run, it is applicable in practice when you have a source user and a target subset of a network, rather than a huge dataset to exhaustively search a match in. Furthermore, our implementation lacks parallelism, though there is a lot of task parallelism when it comes to calculating similarities for features. Since these features are independent, they can be calculated separately, and that would significantly speedup the process.

7 Conclusion

In this section, we will have some final remarks and also go over the proposed and delivered deliverables.

We believe the project has shown that this project serves a proof-of-concept for crawler based profile matching algorithm. Though initially we were planning to use more formal approaches such as finding a dataset online or using the API, using a crawler avenues a lot of opportunities for “on-demand” profile matching, where a user is attempted to be matched to another user in a specific well-defined subset of users. With an on demand crawling capability, the dataset can be formed by need and with these algorithms they can be matched.

Furthermore, the machine learning model we have has been created with just 200 users at hand. With more users, better crawlers and thus more features, a better algorithm is possible: an on-demand and accurate profile matcher!

7.1 Deliverables

Our proposed deliverables, taken directly from our proposal, are as follows:

- In terms of functionality, we will deliver a Python script, that takes a Twitter or Facebook username as input, and matches to a set of candidate users in the dataset from it's respective social network to the other. The results will be shown in an HTML file, for clarity and better UX.
- A final report will be delivered, which includes the performances of implemented algorithms, with metrics such as Accuracy, Precision and Recall and examples of matched users and how they were matched.
- Our source code with clear instructions and documentation will be delivered.

Our current deliveries are:

- The script is working and is included as the source code.
- The final report is written.
- The source code, taken as a release from our GitHub repository, is found within our submission. The code includes the crawler too. Database connection credentials are also written within the code.

We are confident that we have met our proposed deliverables in this project.

References

[1] Zhou, J., Fan, J.: Translink: User identity linkage across heterogeneous social networks via translating embeddings. In: INFOCOM. pp. 2116–2124 (2019)

[2] Halimi, A., & Ayday, E. (2017). Profile Matching Across Unstructured Online Social Networks: Threats and Countermeasures. ArXiv, abs/1711.01815.

[3] Goga, O.: Matching user accounts across online social networks methods and applications. (2014)

[4] Malhotra, A., Totti, L., Meira, W., Kumaraguru, P., Almeida, V.: Studying user footprints in different online social networks. In: Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012 (2012)

[5] Hazimeh, H., Mugellini, E., Abou Khaled, O., Cudre-Mauroux, P.: Social-Matching++: A Novel Approach for Interlinking User Profiles on Social Networks (2017)

[6] Jain, P., Kumaraguru, P., & Joshi, A. (2013). @i seek 'fb.me': identifying users across multiple online social networks. WWW '13 Companion.

[7] W. Chen, H. Yin, W. Wang, L. Zhao, and X. Zhou, "Effective and efficient user account linkage across location based social networks," in ICDE, 2018

[8] Shu, Kai & Wang, Suhang & Tang, Jiliang & Zafarani, Reza & Liu, Huan. (2017). User Identity Linkage across Online Social Networks: A Review. SIGKDD Explorations.18.10.1145/3068777.3068781.

[9] Khaled, O.A.: Linking user profiles in social networks : a comparative review Hussein Hazimeh *, Elena Mugellini and Philippe Cudr ´e-Mauroux 2(4), 333–361 (2017)

[10] Zhang, H., Kan, M., Liu, Y., & Ma, S. (2014). Online Social Network Profile Linkage. AIRS.

[11] Zhang, J., Shao, W., Wang, S., Kong, X., Yu, P.S.: PNA: Partial Network Alignment with Generic Stable Matching. In: Proceedings - 2015 IEEE 16th International Conference on Information Reuse and Integration, IRI 2015 (2015)

[12] Sokhin, T., Butakov, N., & Nasonov, D. (2019, September). User Profiles Matching for Different Social Networks Based on Faces Identification. In International Conference on Hybrid Artificial Intelligence Systems (pp. 551-562). Springer, Cham.

[13] Oana Goga, Patrick Loiseau, Robin Sommer, Renata Teixeira, and Krishna P Gummadi. (2015). On the reliability of profile matching across large online social networks. In KDD

[14] Amos, B., Ludwiczuk, B., Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118, CMU School of Computer Science

[15] https://backlinko.com/social-media-users *Accessed 18.01.2021*

[16] https://www.vultr.com/ *Accessed 18.01.2021*

[17] https://www.vicinitas.io/ *Accessed 18.01.2021*

[18] https://www.selenium.dev/ *Accessed 18.01.2021*

[19] https://developers.facebook.com/docs/graph-api/ *Accessed 18.01.2021*

[20] https://developer.twitter.com/en *Accessed 18.01.2021*

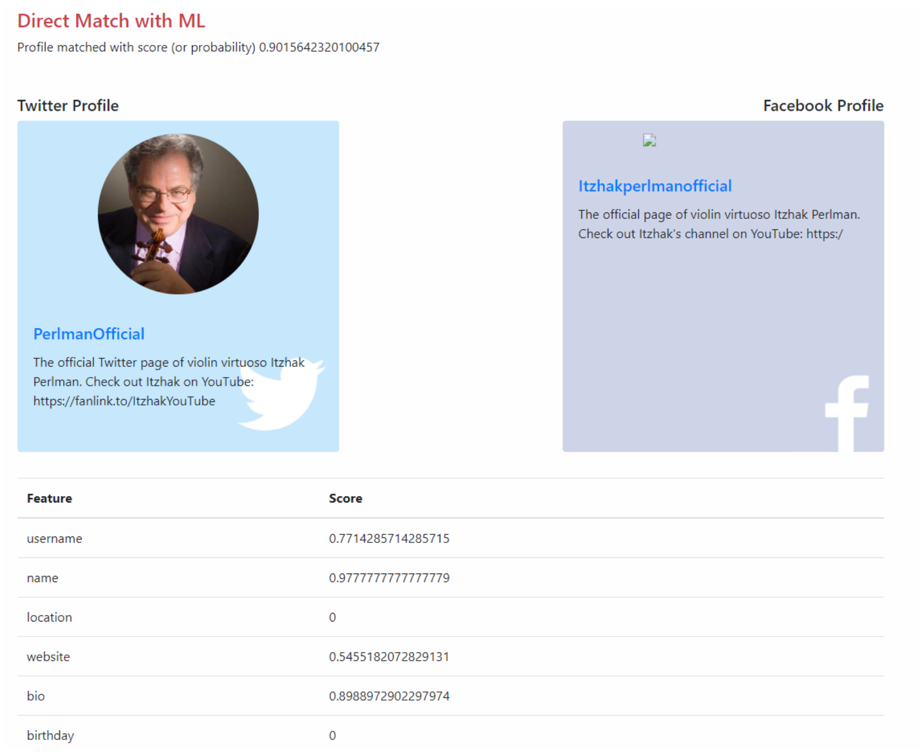
Appendices

A. Metrics[1]

- Accuracy:
$$\frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$
- Precision:
$$\frac{|TP|}{|TP| + |FP|}$$
- Recall:
$$\frac{|TP|}{|TP| + |FP|}$$

[1]Here TP stands for True Positive, TN stands for True Negative, FP stands for False Positive and FN stands for False Negative.

B. UX Module Screenshots[2]



Direct Match with ML

Profile matched with score (or probability) 0.9015642320100457

Twitter Profile

Facebook Profile

PerlmanOfficial

The official Twitter page of violin virtuoso Itzhak Perlman. Check out Itzhak on YouTube: https://fanlink.to/ItzhakYouTube

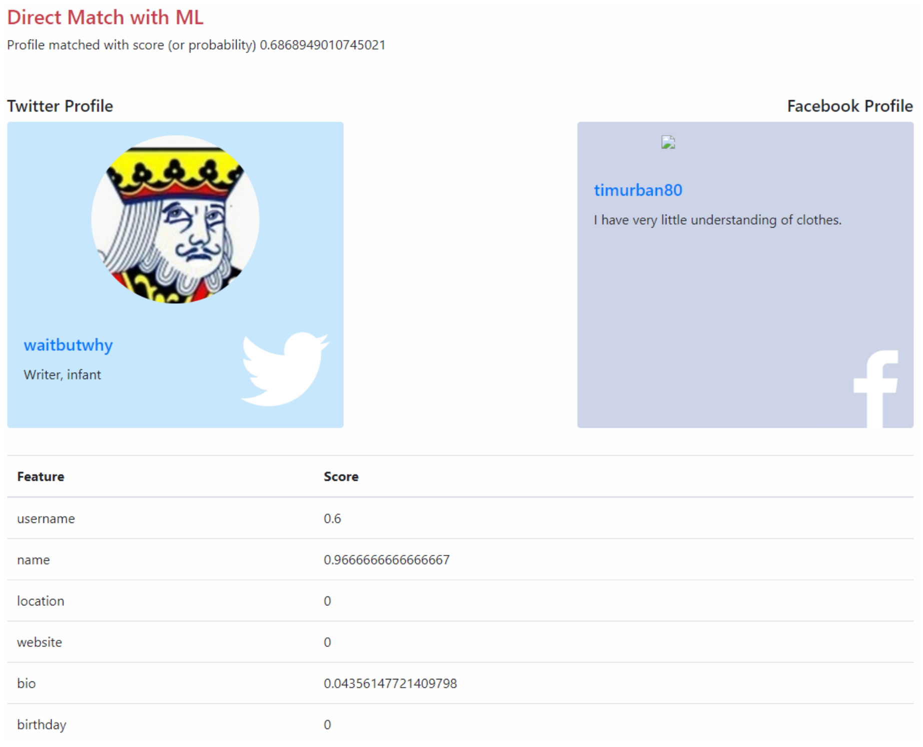
Itzhakperlmanofficial

The official page of violin virtuoso Itzhak Perlman. Check out Itzhak's channel on YouTube: https:/

Feature	Score
username	0.7714285714285715
name	0.9777777777777779
location	0
website	0.5455182072829131
bio	0.8988972902297974
birthday	0

Figure 4: Matching example 1. Many features were found to be similar, and these resulted in a match, with probability 0.9.

[2]The information displayed in this appendix are all public, therefore no private information is being shared.


Direct Match with ML

Profile matched with score (or probability) 0.6868949010745021

Twitter Profile

waitbutwhy

Writer, infant

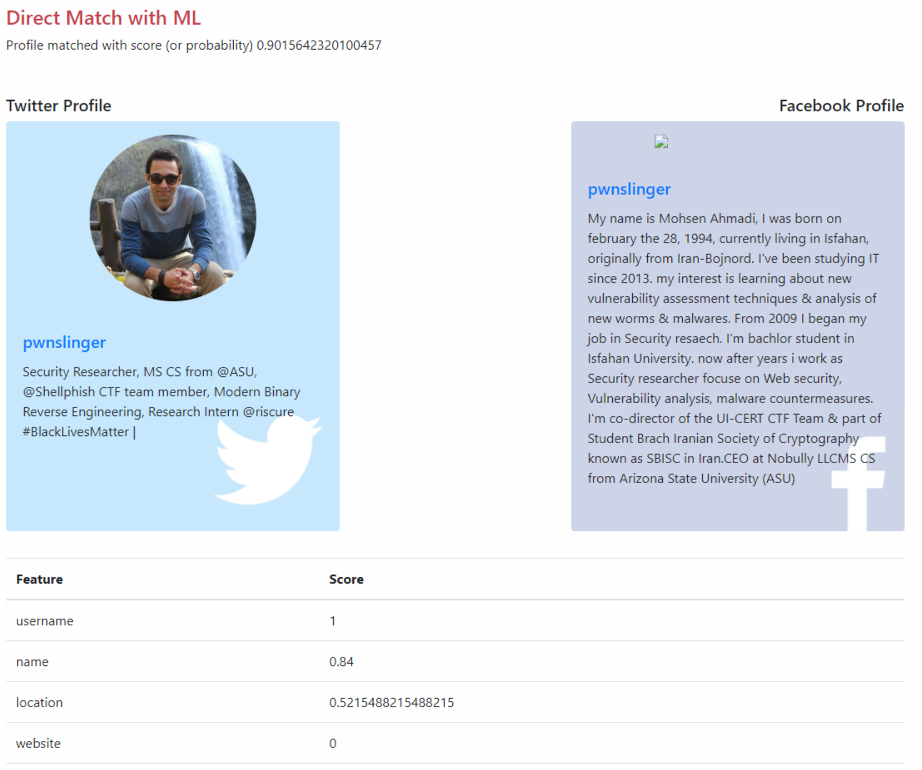
Facebook Profile

timurban80

I have very little understanding of clothes.

Feature	Score
username	0.6
name	0.9666666666666667
location	0
website	0
bio	0.04356147721409798
birthday	0

Figure 5: Matching example 2. Note that the usernames are different, but the name is same in both networks, thus the model was able to match them.



Direct Match with ML

Profile matched with score (or probability) 0.9015642320100457

Twitter Profile

pwnslinger

Security Researcher, MS CS from @ASU, @Shellphish CTF team member, Modern Binary Reverse Engineering, Research Intern @riscure #BlackLivesMatter |

Facebook Profile

pwnslinger

My name is Mohsen Ahmadi, I was born on february the 28, 1994, currently living in Isfahan, originally from Iran-Bojnord. I've been studying IT since 2013. my interest is learning about new vulnerability assessment techniques & analysis of new worms & malwares. From 2009 I began my job in Security reseach. I'm bachlor student in Isfahan University. now after years i work as Security researcher focuse on Web security, Vulnerability analysis, malware countermeasures. I'm co-director of the UI-CERT CTF Team & part of Student Brach Iranian Society of Cryptography known as SBISC in Iran.CEO at Nobully LLCMS CS from Arizona State University (ASU)

Feature	Score
username	1
name	0.84
location	0.5215488215488215
website	0

Figure 6: Matching example 3. Even though the biography is differing, especially due to the length, the probability of matching is still found to be 0.9.