Bachelor thesis
Computer Science

Radboud University

A Comparison of String Distance Metrics on Usernames for Cross-Platform Identification

Author:
Mathis Sackers
s4463455

First supervisor/assessor:
Prof.dr.ir. Arjen P. de Vries
A.deVries@cs.ru.nl

Second assessor:
Maaike H.T. de Boer, Msc.
M.deBoer@cs.ru.nl

June 15, 2017

Abstract

People often use similar usernames across different social media sites. This fact can be used to correlate accounts between different platforms. Since the first mention of this fact in 2009 no research has been done on how to exploit it most efficiently. We showed that ignoring the casing will most definitely improve the matching and we found that Smith-Waterman provides the best metric to match usernames and achieves a success rate of 76%. This implies that earlier work using other string matching metrics could achieve better results by using Smith-Waterman.

Contents

Chapter 1

Introduction

What someone shares on a social media site like Twitter can tell us a lot about that user. But especially when we combine what a user shares across multiple social media sites we can gather much more information about that user [VE16b]. Starting in 2009, from the observation that many users use similar usernames on different platforms [ZL09] much research has been done to identify users across different social networks. Many creative ideas like comparison of writing styles [GLP+13] and the tagging behaviour [IFAB11] have been proposed and tested. These methods of cross-network user-identification are then often compared - as form of a benchmark - to a string-matching method that is applied to the usernames. But, also since 2009, nobody has focused solely on matching usernames.

In this thesis we go back and focus on the interesting fact that usernames of a single user do not seem to differ much between social media platforms. To achieve this we examine eight string-matching metrics on the dataset collected by ETH [VE16a]. This dataset features, amongst other data, accounts from Twitter and Instagram, two of the most popular social media platforms today [1]. These accounts were screened to be authentic and thus provide us with a good starting point.

With our research we hope to use the findings of Zafarani and Liu [ZL09] more practically and to present more data on one of the most telling single features for cross-platform identification, that is also present and publicly accessible on almost all social media platforms.

Specifically we want to answer:

What algorithm presents the highest accuracy?

And can we draw conclusions on the importance of features of a username?

[1] eBizMBA Inc. "Top 15 Most Popular Social Networking Sites — May 2017" eBizMBA.com. `http://www.ebizmba.com/articles/social-networking-websites` (accessed May 27, 2017).

Chapter 2

Related Work

Cross-platform user identification was a novel idea in 2009. Zafarani and Liu claimed to be the first to publish research dealing with that subject matter [ZL09]. After compiling their own data set out of listings on BlogCatalog they found that many users used the exact same username across different networks. Furthermore, even if the usernames were not exactly the same, they would often only be a slightly altered version of their counterparts in different networks. That brought them to use username mapping as an approach to find different profiles of the same user.

Carmagnola and Cena took a look at user-identification in the interest of building better user models [CC09]. They propose a system where data is willingly shared by the web-services themselves to better identify users and thus improve personalization.

Based on this and other work Iofciu, Fankhauser, Abel and Bischoff tried to identify users across Flickr and Delicious based on their tagging activity while also taking usernames into account [IFAB11]. They concluded that, in most cases, identifying users by their username is more effective than using their tagging-behaviour. The combination of the two however, is 35% more effective than only using tagging-behaviour and 8.9% more effective than using Longest Common Subsequence (LCS) on usernames.

Goga et al. found that even meta-data like location, time and writing style, present on Yelp, Twitter and Flickr, can also be used to identify users [GLP+13]. They compared their results to username matching using the Jaro distance and found that the effectiveness is not as good matching from Yelp to Twitter, as it is from Twitter to Flickr. This stems from the fact that many users choose a very different username for Yelp.

Malhotra, Totti, Meira Jr., Kumaraguru and Almeida again took a broad approach and tried to match users based on their profile pictures, their profile's description text and the location [MTMJ+12]. They also matched the usernames using the Jaro distance. The platforms across which this was carried out were LinkedIn and Twitter.

In 2016 Han Veiga and Eickhoff collected a dataset of 850 authentic English-speaking users across Twitter, Instagram and Foursquare [VE16a]. They performed a simple username matching using the Levenshtein distance as an example and were able to attain an accuracy of about 70%.

Aforementioned Goga et al. chose the Jaro distance because of the findings by Cohen, Ravikumar and Fienberg in 2003 [GLP+13] [CRF03]. This publication tested a very broad selection of string-matching metrics and found that Jaro-Winkler is indeed 'surprisingly good'. They did however, run their tests on data sets that do not bear strong resemblance to username sets.

Chapter 3

Research

The string matching metrics have to correctly identify a pair of usernames belonging to the same user across two different social media sites. For example, given the Twitter handle of user u, the metrics should score the Instagram usernames such that u's Instagram username scores highest.

3.1 Method

In total we tested 8 algorithms and - when applicable - a version of the algorithm ignoring the casing in usernames.

We used *Exact Match* as a baseline, as matching two usernames which are exactly the same is trivial.

Most of the other algorithms we compared fall into the category of 'Edit-Distance-Metrics'. They assign a cost to certain forms of transformation from one string to another.

The *Levenshtein distance*, for example, tries to convert one string into another by substituting characters, adding new ones or deleting existing ones. Every such edit is added up and the final result describes the cost of transformation and is thus a measure of string difference.

The *Hamming distance* functions in a very similar way, but only allows for substitution. Normally this means that only strings of equal length can be compared, but we altered it slightly, so that it just adds the difference in length as a penalty.

Longest Common Subsequence, or *LCS* for short, only allows for deletions and additions. Practically it searches for the longest sub-string both strings have in common. Gaps are permitted here.

Smith-Waterman is a version of *LCS*, which penalizes gaps in the sub-string. We have chosen a linear gap penalty.

The *Jaro Distance* and the *Jaro-Winkler Distance* are two more complex Edit-Distance-Metrics and process the number of character transportations needed to edit one string into another. Our implementation already ignores

the casing.

The *Jaccard Index* is a similarity metric that works on sets and is not based on Edit-Distance, but rather 'Token-Based'. It is calculated by dividing the intersection of the two sets by their intersection:

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B}$$

To apply this to our short strings, we implemented two different versions of this algorithm. In the first version, which will subsequently be referred to simply as *Jaccard*, we split the usernames up into their letters and removed duplicates. The string 'username' would thus result in the set $\{u, s, e, r, n, a, m\}$.

The second version, subsequently $Jaccard_2$, splits the strings into tokens of length 2 and then removes duplicates. The string 'user_us' would become $\{us, se, er, r\_, \_u\}$.

3.2 Dataset

To evaluate the above mentioned metrics we chose to use the dataset presented in [VE16a]. It consists of 850 users with profiles across Instagram, Twitter and Foursquare. Foursquare does not let the user pick a username, but instead uses their real names. That is why we chose not to include it in our research.

Furthermore, only the IDs of the accounts are listed in the dataset, due to privacy concerns and the Terms of Service of the platforms in question. This means that we had to scrape the data ourselves from the sites. Our queries to Twitter and Instagram revealed, that about 9% of account-pairs were not available at the time of access. These stem from about 1.4% of missing Twitter accounts and 8.2% Instagram accounts. We still included the usernames of accounts which had no counterpart in the other network as possible guesses for the algorithms.

3.3 Metrics

To measure the performance of the algorithms we employed the same two metrics Iofciu et al. employed when comparing their username matching metrics: *MRR* and *S@k* [IFAB11]. However we do not apply their method of handling ties since it penalizes them very harshly. So much so that there are cases in which taking the worst case (where the correct pair appears at the bottom of the tied scores) would receive a better score than it does in the system proposed in [IFAB11].

MRR (Mean Reciprocal Rank) gives an average of the position the correct result appears at. It is calculated as the average multiplicative inverse

of the rank the correct result appears at:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Here $|Q|$ denotes the number of account pairs and $rank_i$ the rank at which the ith account appears at. The MRR always lies between 0 and 1. An MRR of 1 implies that the correct result averages on position 1, while an MRR close to 0 implies that the result on average can be found near the bottom of the ranking. To handle the ties between multiple results we chose the average position the correct result would appear at in a random ordering of tied scores.

S@k denotes the Success at rank k. The chance of finding the correct result at the specified rank is calculated like this:

$$S@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} min\left(max\left(\frac{(k+1) - rank_i}{ties_i}, 0\right), 1\right)$$

Again, $|Q|$ and $rank_i$ indicate the number of pairs and the rank at which the correct one appears respectively. But instead of averaging the position, we use the number of guesses ranked at this rank $ties_i$ to calculate the probability of the correct guess appearing in the top k results. The intermediate results are also clamped to be in the range of $[0, 1]$.

Furthermore, we used a two-sided *sign test* on a few select algorithm pairs to evaluate if their results differ significantly. To do this we compared the average position the individual accounts would be ranked at and used a significance level of p = 0.05.

Chapter 4

Results

Table 4.1 presents our results for matching between Instagram usernames and Twitter 'handles', which are their form of unique usernames. The first row already produces interesting results. The 'Exact Match'-algorithm should only place the *exact* match at the top of the rating and thus S@1 and S@3 should both be the same! Where does the slight difference come from? This has to do with our approach to handling ties. An exactly matching username will indeed produce a ranking where the correct guess stands at the top and S@1, S@3 and S@10 will all be 1. But if the algorithm does not find any name that exactly matches its candidate no name will be awarded points and thus all possibilities will be tied at first place. In a random ranking between ties the chances of the correct result appearing at the top position (S@1) is of course very slim, but still taken into account. The chance for the correct result to appear in the top three results (S@3) is thrice as high and greater again for S@10.

The scoring of the individual algorithms is symmetrical, meaning that the ordering of the two input strings does not matter. Yet, we still see a slight difference in most our values between matching from Instagram to Twitter versus the other way around. This, of course, stems from the fact that the matching from Instagram to Twitter evaluates one name on Instagram against all names on Twitter. Interestingly most metrics vary only by about 1%, while *Jaccard* differs by nearly 5% at S@1. The sign test reveals that this difference is actually a very significant one ($p < 0.0001$).

Furthermore, it is obvious that all implementations ignoring case perform a bit better than their case-sensitive counterparts. The biggest gain in performance - by far - can be seen in both *Jaccard* and *Exact Match*. Seeing *Exact Match* perform so much better will probably stem from the fact that Instagram does not allow upper case letters in their usernames. This is probably also the cause for *Jaccard*'s performance gain.

Another observation is that *Jaccard* and $Jaccard_2$ both improve a lot from S@1 to S@10 compared to the edit-distances.

While *Jaccard* still performs worst in the set, token-based approaches should not be scrapped completely, as its counterpart $Jaccard_2$ actually performs better than most edit-distance metrics. And the sign test shows that the null hypothesis between $Jaccard_2\_ic$ and *Smith-Waterman_ic* can not be rejected ($p = 0.9434$).

Which brings us to our best metric: the case-ignoring *Smith-Waterman* algorithm, which correctly identifies about 76% of the users. This interestingly stands in conflict with the findings of [ZL09] who found both *LCS* and *Levenshtein* to perform better than *Smith-Waterman*. And even though the difference between *LCS_ic* and *Smith-Waterman_ic* is not significant ($p = 0.1208$), the difference to *Levenshtein_ic* very much is ($p = 0.0002$).

	Instagram to Twitter				Twitter to Instagram			
Strategy	MRR	S@1	S@3	S@10	MRR	S@1	S@3	S@10
Exact Match	0.199	0.296	0.298	0.304	0.199	0.296	0.298	0.304
Exact Match_ic	0.384	0.574	0.575	0.579	0.383	0.574	0.575	0.579
Levenshtein	0.471	0.692	0.709	0.725	0.470	0.691	0.707	0.728
Levenshtein_ic	0.490	0.723	0.735	0.753	0.487	0.718	0.733	0.750
Hamming	0.444	0.654	0.671	0.682	0.443	0.651	0.666	0.682
Hamming_ic	0.461	0.681	0.693	0.704	0.456	0.674	0.683	0.697
LCS	0.454	0.657	0.692	0.722	0.468	0.681	0.712	0.743
LCS_ic	0.495	0.727	0.744	0.769	0.496	0.729	0.749	0.769
Smith-Waterman	0.484	0.700	0.737	0.762	0.489	0.710	0.746	0.768
Smith-Waterman_ic	0.516	0.760	0.780	0.795	0.516	0.754	0.782	0.798
Jaro	0.494	0.725	0.745	0.753	0.493	0.727	0.741	0.753
Jaro-Winkler	0.493	0.725	0.738	0.757	0.494	0.726	0.742	0.755
Jaccard	0.325	0.398	0.513	0.602	0.358	0.447	0.559	0.664
Jaccard_ic	0.439	0.590	0.693	0.730	0.426	0.568	0.674	0.717
$Jaccard_2$	0.484	0.696	0.736	0.761	0.493	0.719	0.744	0.762
$Jaccard_2$_ic	0.512	0.748	0.772	0.788	0.513	0.749	0.772	0.794

Table 4.1: Matching Instagram usernames and Twitter handles.
'_ic' denotes the case ignoring version of the algorithm.

Chapter 5

Conclusions and Future Work

In this thesis we investigate the performance of a number of string-matching-metrics. To achieve this we evaluate them on a dataset of about 850 account pairs across Twitter and Instagram. Specifically we were looking for (1) the algorithm with the highest accuracy and (2) the importance of different features in a username. We can conclude from our experiments that (1) Smith-Waterman is the best-performing edit-distance-metric in our selection, and (2) that while casing does not play an important role in usernames, and the letters contained in the names are not telling us much about the users, looking at recurring subsequences or even just pairs of letters gives us good results. This implies that maybe greater accuracies in cross-platform identification, than the ones demonstrated in earlier literature, are possible, as some of them combined different methods with weaker forms of username matching.

For future work it would be interesting to see a combination of the token-based *Jaccard* approach and an edit-distance like *Smith-Waterman*. The dataset we used also contains much more additional information that could be used to enhance the user matching, like their Tweets or Instagram posts. Furthermore, we are curious to see the discrepancy between our work and the findings of [ZL09] investigated.

Acknowledgements

I would like to thank the following people for their guidance and their support in completing this thesis. In alphabetical order:

Arjen P. de Vries,
Björn Sackers (father),
Christopher Strucks,
Inta Sackers (mother),
Jan Feldmann,
Julian Teufel

Bibliography

[CC09] Francesca Carmagnola and Federica Cena. User identification for cross-system personalisation. *Information Sciences*, 1(179):16–32, 2009.

[CRF03] W. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *In International Joint Conference on Artificial Intelligence (IJCAI) 18, Workshop on Information Integration on the Web*, 2003.

[GLP+13] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 447–458, New York, NY, USA, 2013. ACM.

[IFAB11] Tereza Iofciu, Peter Fankhauser, Fabian Abel, and Kerstin Bischoff. Identifying users across social tagging systems. In *ICWSM*, 2011.

[MTMJ+12] Anshu Malhotra, Luam Totti, Wagner Meira Jr., Ponnurangam Kumaraguru, and Virgilio Almeida. Studying user footprints in different online social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ASONAM '12, pages 1065–1070, Washington, DC, USA, 2012. IEEE Computer Society.

[VE16a] Maria Han Veiga and Carsten Eickhoff. A cross-platform collection of social network profiles. *CoRR*, abs/1607.03274, 2016.

[VE16b] Maria Han Veiga and Carsten Eickhoff. Privacy leakage through innocent content sharing in online social networks. *CoRR*, abs/1607.02714, 2016.

[ZL09] Reza Zafarani and Huan Liu. Connecting corresponding identities across communities. In *International AAAI Conference on Web and Social Media*, 2009.

Appendix A

Algorithms

Here we want to present additional information needed to configure the algorithms presented in Section 3.1 to work in the way they did for us.

Jaro and Jaro-Winkler

For both Jaro and Jaro-Winkler we employed a scaling of 0.1.

Smith-Waterman

The Smith-Waterman is configured by a substitution matrix and a gap penalty. We chose this substitution matrix:

$$S(x, y) = \left\{ \begin{array}{ll} +3, & x = y \\ -3, & x \neq y \end{array} \right\}$$

As mentioned, we are using a linear gap penalty, concretely this one:

$$W_k = kW_1$$

$$W_1 = 2$$

Appendix B

Dataset Remarks

We mentioned in Section 3.2, that about 9% of the accounts were not accessible at the time of writing. Concretely, we were missing the accounts belonging to the following ids:

Missing Twitter IDs

109172969, 2900512275, 3036621918, 312948791, 381127120, 384299434, 4657587022, 49649609, 1429144974, 1623688176, 54533228, 773528706

Missing Instagram IDs

144105195, 600002898, 410336439, 52195935, 173951616, 24768895, 5417659, 176731678, 763470867, 33300847, 369734304, 9604228, 9376383, 52033824, 12438017, 174429883, 311002729, 260867560, 389876472, 11230594, 404060, 174002037, 19712477, 3334755, 304774881, 246947747, 236139652, 43542850, 232095312, 2117117604, 31264833, 45605371, 218551342, 250603554, 32799110, 143710160, 332121968, 207934782, 41044905, 530869490, 32833211, 1824763818, 237351, 2574638, 226751723, 231670770, 2338857, 12241611, 187662790, 31234239, 16394107, 29014694, 270367408, 1275342750, 1215127498, 211898978, 251136385, 31346994, 48194187, 174435864, 31152722, 146013228, 321623520, 41272419, 866354408, 51913460, 23688245, 408551582, 1792048, 36046998