The 2016 AAAI Fall Symposium Series:
Privacy and Language Technologies
Technical Report FS-16-04

An Analysis of Blocking Methods for Private Record Linkage

Brian Etienne
Univ. of Massachusetts Dartmouth
Dartmouth, MA 02747

Michelle Cheatham* and **Pawel Grzebala**
Wright State University
Dayton, OH 45324

Abstract

The field of Big Data Analytics has led to many important advances; however, these analyses often involve linking databases that contain personal identifiable information, and this raises privacy concerns. The sensitive information of individuals can be stolen by attackers and identities can be exploited. The field of Private Record Linkage seeks to mitigate these risks. Our previous work in this area has established the effectiveness of a fuzzy string similarity metric for linking records while the data within those records remains encrypted both on disk and in memory. The work presented here further contributes to the PRL field by analyzing the performance of various blocking methods on encrypted names that attempt to optimize the number of comparisons between the queries and database while maintaining a high rate of accuracy.

Introduction

Privacy is very difficult to maintain with the capabilities of technology today. Consider a database that holds personal identifiable information such as names, social security numbers, credit card numbers, addresses, and more. This database can easily be hacked and the confidential data can land in the hands of the wrong person if the right precautions are not taken. Situations like this have occurred many times before. For example the Target hack in 2013 (tar), the attack on United States Postal Service in 2014 (USP), and the Blue Cross Blue Shield hack in 2015 (blu). In all these cases, confidential information of many people were stolen. The goal of this work is to provide a solution to this problem.

This project investigates a scenario in which a company that maintains a database containing information about individuals wishes to allow customers who have paid for access to the database to query it. Unlike most work related to PRL, the database owner and the querying party trust one another, and neither is concerned with the other learning what information is in the database and what records are being queried for. However, the database owner is concerned about an unauthorized party gaining access to the computer on which the database is stored and so desires to keep the information within the database encrypted both on disk and in memory. A symmetric encryption key is shared with authorized parties, who use that key to encrypt their query and it is the encrypted values that are used to query the database.

Our previous work on this topic has shown that computing the Dice similarity metric between sets of encrypted bi-grams of the two strings to be compared yields good performance for this use case (Grzebala and Cheatham 2016). For example, the name "John" is converted into the set of bi-grams ["Jo" "oh" "hn"]. Each bi-gram is encrypted using the AES symmetric encryption algorithm and the set of encrypted bi-grams is used to query the database. The degree of similarity between the two names being compared is determined by the Dice Similarity Coefficient, which calculates the degree of overlap between the two sets of encrypted bi-grams. Q-grams (in this case q=2) are a suitable similarity metric for fuzzy matching because a misspelling or mistyping of a name will only change a few of the q-grams, and there will therefore still be a high degree of overlap.

That previous work compared the name being queried for to the name field for every record in the database using the Dice similarity on the bi-grams. This is a fairly expensive process. In this work we focus on enhancing the performance and scalability of the record linkage by using blocking methods to keep the number of expensive comparisons to a minimum by discounting records with names that clearly do not match the query, while maintaining high precision and recall. Specifically, this work addresses the following:

1. Which blocking method is most suitable when querying first and last names?
2. Does the optimal threshold change if blocking is used?
3. Does the utility of blocking change if mistakes such as typos or OCR errors are present in the data?
4. Does the utility of blocking change as the number of mistakes in the data changes?

Our goal is to establish a set of guidelines for PRL practitioners who wish to use blocking techniques. Our focus here is on string data; we plan to consider numeric data in future work.

Related Work

There have been numerous approaches to solving the general problem of record linkage based on person names. A

*Corresponding author: michelle.cheatham@wright.edu

comprehensive overview of several name matching techniques was provided by Snae in (Snae 2007). Snae describes four different types of name matching algorithms and compares them in terms of accuracy and execution time. The results indicated that there is no single best method for name matching and the proper choice depends on the specific application needs. This work doesn't take into consideration the important aspect of our study, which is linking records while keeping them encrypted. That topic is addressed somewhat in (Dusserre, Quantin, and Bouzelat 1994), which proposes a secure one-way hash to protect the string fields for PRL, but this approach only supports exact matching of strings. Churches and Christen proposed the use of encrypted Soundex or Metaphone phonetic encodings of names for private record linkage (Churches and Christen 2004). These comparisons are still based on exact match, but it is the exact match on the phonetic spelling of the names, which mitigates the impact of common misspellings. In the same paper, they also propose encrypted q-grams as a viable approach for fuzzy string matching of encrypted strings. Our work in (Grzebala and Cheatham 2016) empirically evaluated the efficacy of these approaches on a realistic dataset and found only the q-gram approach to be viable. This description is necessarily abbreviated, but a more in-depth review can be found in (Christen 2012a).

Blocking (also called indexing) methods attempt to group similar strings together such that not every record needs to be compared when responding to a query. These techniques have been studied for a long time in the plaintext (unencrypted) context. A survey can be found in (Christen 2012b). In general, privacy-preserving blocking methods have received much less attention from researchers than have privacy-preserving similarity metrics. The primary work on privacy-preserving blocking methods of which we are aware is (Al-Lawati, Lee, and McDaniel 2005), which proposes a third-party based approach using secure hashing of TF-IDF weight vectors. The work presented here differs from previous efforts in that in analyzes the performance of a wider variety of blocking methods in the presence of many different types of corruptions commonly present in databases, as described in the following section.

Approach

The blocking methods explored here are Soundex, Metaphone, prefix and suffix. Soundex and Metaphone are phonetic encoding algorithms. While Soundex was shown to perform poorly as a string similarity metric for PRL (Grzebala and Cheatham 2016), it is quick to compute and could be useful blocking key. Both Soundex and Metaphone break the database into blocks containing similar sounding names. Every name in the database is encoded by the phonetic algorithm and then encrypted using the AES symmetric encryption algorithm. All names with the same encrypted value of the phonetic algorithm are organized into a block. Prefix based blocking uses the first n characters of a name as the blocking key, where n is a parameter to the blocking method. The prefixes are encrypted. The suffix approach is similar, except it uses the last n characters of the name.

The data used for this project was generated using the GeCo Data Generator (Tran, Vatsalan, and Christen 2013). This tool is capable of generating PII such as names, phone numbers, social security numbers, credit card numbers, etc. that is based on real world data. GeCo is also capable of corrupting data in various ways: keyboard edits (replacing a character with one nearby on the keyboard), character edits (adding, deleting, or transposing characters), optical character recognition (OCR) edits, and phonetic edits. This tool also allows the user to set the number of corrupted records, the number of corruptions applied to a record, and the number of modifications applied to each record attribute.

GeCo was used to generate a dataset with 10,000 records. Each record contains a first name, last name, gender, social security number, and credit card number. This dataset was then corrupted using each of the four corruption types described above, as well as a mix of all four types of corruption, for a total of five variations. Two sets of each variation were created: one that contained one modification per attribute and a second that contained two modifications per attribute. Each of the 10 corrupted datasets contained 10% corruption, i.e. 1,000 records were randomly corrupted.

A query for each name in the dataset was simulated by attempting to match the uncorrupted record to its corrupted version. The records in the uncorrupted dataset were blocked according to the method currently being analyzed. Then the blocking key for each record in the corrupted dataset was computed using the same blocking algorithm. The corrupted record was then compared to each of the uncorrupted records with the same blocking key. This comparison was done based on bi-grams of the names using the Dice Similarity Coefficient, as described in Section 1. If the similarity value was equal to or exceeded a specified threshold value, the names being compared were considered a match and the records were linked. If the two records did indeed have the same name, the match was considered a true positive. Otherwise, it was considered a false positive. If the similarity value did not exceed the specified threshold for the matching record, the case was considered a false negative. This analysis was done for both first and last names.

Analysis

The soundex, metaphone, prefix, and suffix blocking methods were evaluated based on the number of comparisons performed, recall, and precision. In this section we analyze the performance of the different blocking methods with respect to the research questions specified in Section I.

Observation 1: *Prefix blocking is preferable for both first and last names*

Blocking Method	Length	Recall	Comparisons	Recall/Comparisons
Soundex		0.9462	433325	2.18358E-06
Metaphone		0.9462	437611	2.16219E-06
Prefix	1	0.9463	6532916	1.44851E-07
	2	0.9463	1749498	5.40898E-07
	3	0.9463	408598	2.31597E-06
	4	0.9462	201421	4.69762E-06
Suffix	1	0.9463	14551753	6.503E-08
	2	0.9463	2243380	4.21819E-07
	3	0.9463	522142	1.81234E-06
	4	0.9463	219204	4.31698E-06

Table 1: Recall/Comparison Ratio calculated with a threshold of 0.95

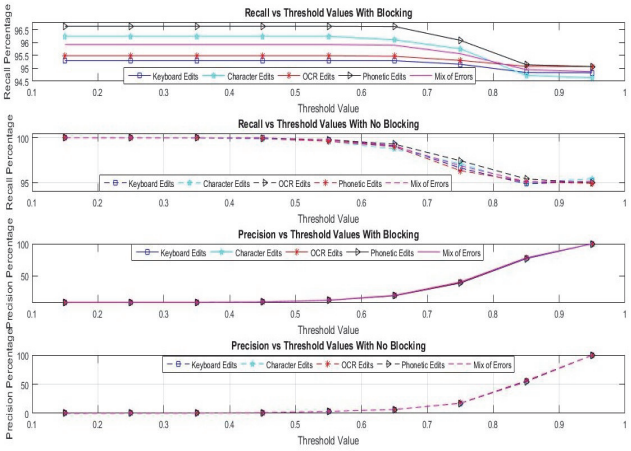

Figure 1: This figure shows the recall and precision for first names against different threshold values for the corrupted datasets containing one modification per attribute

Prefix blocking with length 4 was most suitable for linking records on both first and last names. This was determined by calculating the ratio of recall to number of comparisons as shown in Table I. This ratio represents the number of successfully linked records per comparison. This test was done for every corrupted dataset, and prefix blocking with length 4 always resulted in the highest recall/comparison ratio. It effectively doubled the performance of Soundex and Metaphone. The performance of suffix blocking was closer but still inferior. As a result, prefix blocking of length 4 was used for all further tests.

Observation 2: *Prefix blocking does not change the optimal threshold*

The effect of threshold value on precision and recall was also explored. Figure 1 shows the result of this experiment for first names; the results for last names show a similar pattern. The upper graph in each pair reflects when prefix blocking of length 4 was used, and the lower graph when no blocking was used. As expected, higher thresholds lead to lower recall and higher precision for all corruption types, whether or not blocking is used. Also notable is that, while recall is slightly higher and precision is slightly lower without blocking, the relative precision and recall at different thresholds are the same whether or not blocking is used. To find the optimal threshold value, the f-measure was calculated. The threshold value resulting in the highest f-measure was 0.95. This was the case regardless of whether or not blocking was used and regardless of corruption type.

Observation 3: *The impact of different corruption types depends on the nature of the string fields being linked*

We now look more closely at the performance in the presence of different types of corruption. We again use prefix blocking with length 4 and a threshold of 0.95. Each record has at most one modification. Precision at this threshold is perfect in the case of first names and almost perfect in the case of last names (Figure 2). We therefore turn our attention to recall and the number of comparisons required.

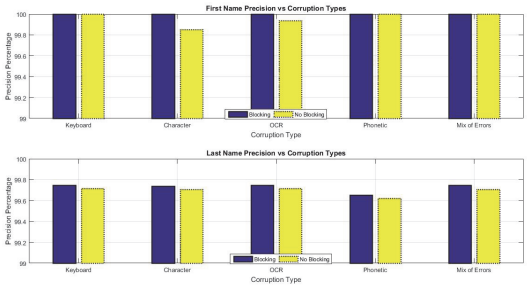

Figure 2: This figure shows the precision percentages for first and last names when using prefix blocking of length 4 and no blocking at a threshold of 0.95 on the corrupted datasets containing one modification per attribute.

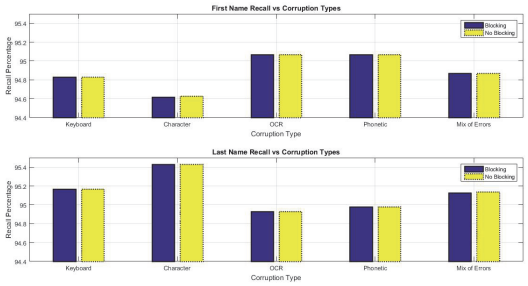

Figure 3: This figure shows the recall percentages for first and last names when using prefix blocking of length 4 and no blocking at a threshold of 0.95 on the corrupted datasets containing one modification per attribute.

Analyzing recall (Figure 3) reveals something interesting: the most challenging type of corruption depends on the nature of the string field being linked. In the case of first names, the recall was highest for phonetic and OCR edits and lowest for character edits, while for last names the exact opposite was true. This is not an impact of blocking but rather due to the characteristics of first and last names. First names are on average shorter than last names, and corruptions that can change the number of bi-grams (e.g. phonetic changes like ai $\rightarrow$ a or OCR errors like d $\rightarrow$ cl) therefore have a larger effect on first names than last.

Unlike recall, the *relative* number of comparisons made using prefix blocking of length 4 did not vary based on corruption type – the pattern in both the top and bottom of Figure 4 is the same. However, the number of comparisons made did vary greatly between first and last names – the number of comparisons for last names was an order of magnitude smaller. This is again due to the average length of first versus last names, combined with the fact that there is more diversity among last names than among first names. Using the first four letters of a last name as a blocking key therefore creates more diverse blocks to organize names than it would for first names which are both more common and shorter. It is these situations in which blocking is most useful.

Observation 4: *Blocking does not change performance even*

Figure 4: This figure shows the number of comparisons for first and last names when using prefix blocking of length 4 and a threshold of 0.95 on the corrupted datasets containing one modification per attribute.

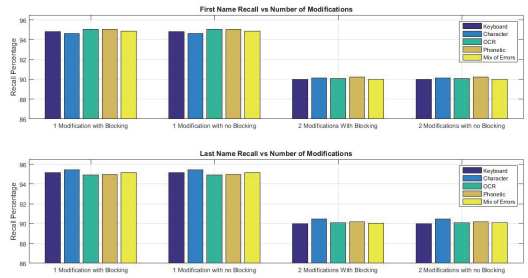

Figure 5: This figure shows the recall percentage for first and last names with 1 and 2 modifications when using prefix blocking of length 4 and no blocking at a 0.95 threshold.

when more mistakes are present, but there is less flexibility

Figures 5 and 6 show that, as expected, recall and precision are significantly lower if names have two mistakes rather than only one. The drop is due to more false negatives. Of particular interest here is that the results using blocking and those without are essentially the same – no extra price is paid in the two corruption per name case when blocking is applied. Meanwhile, Figure 7 shows that blocking saves even more comparisons when names contain more mistakes. This is because any mistake in the first four characters of a name will cause the record to be put in a different block. Since many names have similar origins and phonetic roots, making random changes such as character, keyboard, or OCR edits significantly increases the numbers of blocks, thereby reducing the number of names in each block.

At first this seems promising – blocking can reduce the number of comparisons without sacrificing accuracy, even in the presence of multiple errors per record. However, the recall in the non-blocking case can be raised (at the expense of further worsening precision) by increasing the threshold value. We discuss this in more detail in (Grzebala and Cheatham 2016). This flexibility is useful for applications in which recall is more important than precision. No such flexibility is possible when prefix blocking is used because exact match of the prefix (i.e. a 1.0 threshold) is always used when locating a record's block.

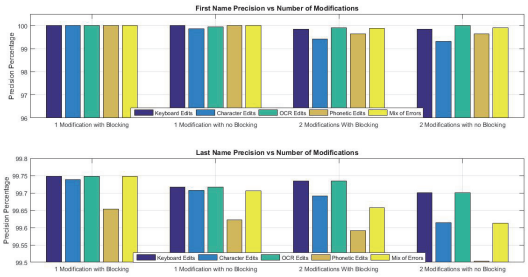

Figure 6: This figure shows the precision percentage for first and last names with 1 and 2 modifications using prefix blocking of length 4 and no blocking at a 0.95 threshold.

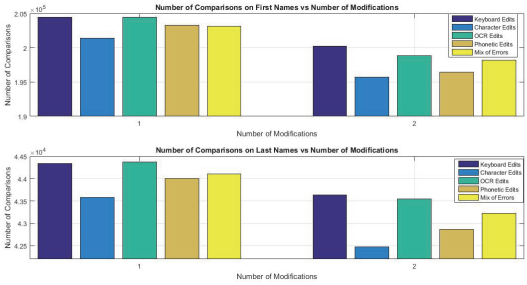

Figure 7: This figure shows the number of comparisons of first and last names with 1 and 2 modifications when using prefix blocking of length 4 and a threshold of 0.95.

Conclusions

This work explores the performance of various blocking techniques that can be used on encrypted strings to improve the scalability of name-based PRL applications. Several guidelines for practitioners are suggested: prefix blocking of length 4 is the best of the approaches analyzed for both first and last names, the threshold for the underlying similarity metric does not need to be modified when prefix blocking is used, and the number of comparisons is reduced by three orders of magnitude for first names and four for last names. Practitioners should be aware that the impact of different types of mistakes varies when blocking is used and there is less flexibility to trade recall for precision.

Acknowledgments: This work was partially supported by the NSF Research Experience for Undergraduates Site Award "Cyber Security Research at Wright State University" (CNS 1560315) and by the Lexis Nexis corporation.

References

Al-Lawati, A.; Lee, D.; and McDaniel, P. 2005. Blocking-aware private record linkage. In *Proceedings of the 2nd international workshop on Information quality in information systems*, 59–68. ACM.

Bluecross blueshield hacked; 10.5m patients affected. http://www.washingtontimes.com/news/2015/sep/10/excellus-

bluecross-blueshield-hacked-105m-patients/. Accessed: 2016-07-22.

Christen, P. 2012a. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Christen, P. 2012b. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering* 24(9):1537–1555.

Churches, T., and Christen, P. 2004. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making* 4(1):1.

Dusserre, L.; Quantin, C.; and Bouzelat, H. 1994. A one way public key cryptosystem for the linkage of nominal files in epidemiological studies. *Medinfo. MEDINFO* 8:644–647.

Grzebala, P., and Cheatham, M. 2016. Private record linkage: Comparison of selected techniques for name matching. In *International Semantic Web Conference*, 593–606. Springer.

Snae, C. 2007. A comparison and analysis of name matching algorithms. *International Journal of Applied Science. Engineering and Technology* 4(1):252–257.

Target credit card hack: What you need to know. http://money.cnn.com/2013/12/22/news/companies/target-credit-card-hack/. Accessed: 2016-07-22.

Tran, K.-N.; Vatsalan, D.; and Christen, P. 2013. Geco: an online personal data generator and corruptor. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2473–2476. ACM.

China suspected of breaching u.s. postal service computer networks. http://www.washingtonpost.com/news/federal-eye/wp/2014/11/10/china-suspected-of-breaching-u-s-postal-service-computer-networks/. Accessed: 2016-07-22.