To appear in: Proceedings of Statistics Canada Symposium 2010
Social Statistics: The Interplay among Censuses, Surveys and Administrative Data

Private Record Linkage with Bloom Filters

Rainer Schnell, Tobias Bachteler, and Jörg Reiher[1]

Abstract

In many record linkage applications, identifiers have to be encrypted to preserve privacy. Therefore, a method for approximate string comparison in private record linkage is needed. We describe a new method of approximate string comparison in private record linkage. The main idea is to store q-grams sets derived from identifier values in Bloom filters and compare them bitwise across databases. This exploits the cryptographic features of Bloom filters while nevertheless allowing the calculation of string similarities. We show that the proposed method compares quite well to evaluating string comparison functions with plain text values of identifiers.

Key Words: Privacy-preserving Record Linkage; Private Record Linkage; Bloom Filter

1. Introduction

Combining multiple databases with disjunctive or additional information on the same person is occurring increasingly throughout social research. The availability of large databases and unique person identifier (ID) numbers has made widespread use of record linkage possible. But in many research applications not all databases contain a unique ID number. In such situations, probabilistic record linkage is most frequently applied for the identification of matching record pairs. However, in many applications the identifiers have to be encrypted due to privacy concerns. Linking with encrypted identifiers can result in biased estimates. Although some intriguing approaches for private record linkage have been proposed in the literature, each involves either very high computing demands or high rates of false positives or false negatives. Hence we developed a new procedure that addresses these problems.

1.1. The Private Record Linkage Problem

Databases of people usually contain identifiers like surnames, given names, date of birth, and address information. Distribution of scientific files containing such information is legally restricted in most countries. The problem of finding records that represent the same individual in separate databases without revealing the identity of the individuals is called the “private record linkage”, “blind data linkage”, or “privacy-preserving record linkage” problem.

An obvious solution for private record linkage seems to be the encryption of the identifiers with a standard cryptographic procedure. An example is the Keyed-Hash Message Authentication Code (HMAC). However, since these protocols require exact matching of identifiers, they do not tolerate any errors in these identifiers. Applying probabilistic record linkage (Herzog et al., 2007) improves the situation considerably since it does not require exact agreement in all (or even most) identifiers. Rather, agreements in better differentiating identifiers might balance disagreements in other identifiers. However, using string similarity functions within a probabilistic record linkage system will improve the linkage quality considerably. In addition, since records with variations of identifiers may have different characteristics than records with exact matching identifiers, restricting the linkage in this manner is not an option. Therefore, a method for approximate string matching in private record linkage is required.

[1] Rainer Schnell (rainer.schnell@uni-due.de), Tobias Bachteler (tobias.bachteler@uni-due.de), Jörg Reiher (joerg.reiher@uni-due.de), University of Duisburg-Essen, Lotharstr. 65, D-47057 Duisburg, Germany.

1.2 Previous Solutions

Some protocols rely on exact matching of encrypted keys based on phonetically transformed identifiers by a third party. In the proposal of Bouzelat et al. (1996) identifiers are transformed according to phonetic rules and subsequently encrypted with a oneway hash function. The hash values are transferred to a third party who performs exact matching on the resulting hash values. Despite exact matching, the linkage allows for some errors in identifiers, because hash values of phonetic encodings are matched. However, string comparison using phonetic encodings usually yields more false positive links than string similarity functions (Camps and Daude, 2003).

Churches and Christen (2004) suggested a protocol based on hashed values of sets of consecutive letters (*q*-grams). For each string, the database holders *A* and *B* create for each record the power set of the *q*-grams of their identifiers. Each subset of the power set is hashed by an HMAC algorithm using a common secret key of the database owners. *A* and *B* form tuples containing the hash values, the number of *q*-grams in the hashed subset and the total number of *q*-grams and an encryption of the identifiers to a third party *C*. To calculate the string similarity between two strings *a* and *b*, *C* computes a similarity measure based on the information in the tuples. *C* is able to determine a similarity measure of *a* and *b* by selecting the highest similarity coefficient of the tuples associated with *a* and *b*. Apart from an increase of computational and communication costs (Verykios et al., 2009), the protocol is prone to frequency attacks on the hashes of the *q*-gram subsets with just one *q*-gram (Trepetin, 2008).

In the protocol of Scannapieco et al. (2007), two data holders, holding lists of names, build an embedding space from random strings and embed their respective strings therein using the SparseMap method (Hristescu and Farach-Colton, 1999). Then, each data holder sends the embedded strings to a third party who determines their similarity. To create the embedding space, data holder *A* generates *n* random strings and builds *z* reference sets from them. Next, *A* reduces the number of reference sets by the greedy resampling heuristic of SparseMap to the best $k < z$ reference sets. These *k* reference sets are used to embed the names in a *k*-dimensional space. The coordinates for a given name are approximations of the distances between the name to the closest random string in each of the *k* reference sets in terms of the edit distance. As a result, for each name *A* receives a *k*-dimensional vector. After receiving the *k* reference sets from *A*, *B* embeds his names in the same way. Finally, both data holders send their vectors to a third party, *C*, who compares them using the standard Euclidean distance between them.

1.3 Objectives of Article

The previous solutions transform the identifiers in a manner that allows consideration of string similarities in a probabilistic record linkage procedure despite encrypting them. However, most existing protocols have a number of problems, e.g. they involve very high computing demands or high rates of classification errors. Therefore, we developed a new method for the calculation of the similarity between two encrypted strings for use in probabilistic record linkage procedures.

2. Methods

The core problem of a private record linkage protocol is the calculation of the similarity of two encrypted strings. We suggested the use of Bloom filters for solving this problem (Schnell et al., 2009). A Bloom filter is a data structure proposed by Bloom (1970) for checking set membership efficiently. Bloom filters can also be used to determine whether two sets approximately match (Jain et al., 2005).

2.1 Bloom filter

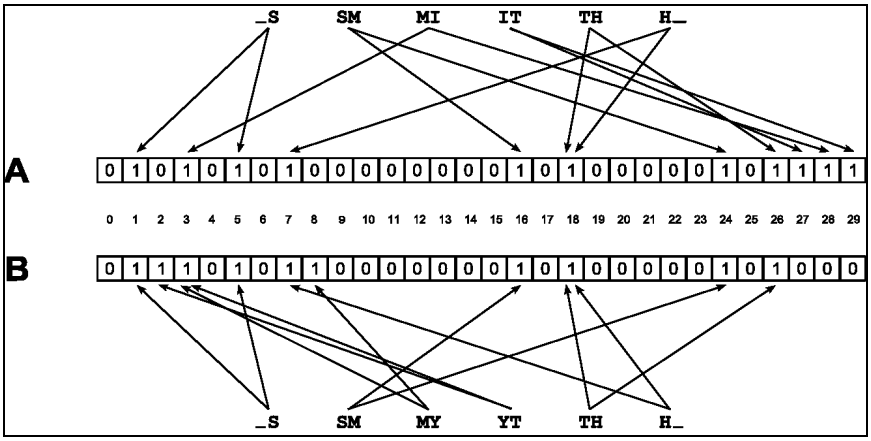
A Bloom filter is a bit array of length *l* with all bits initially set to 0. Furthermore, *k* independent hash functions $h_1, \ldots, h_k$ are defined, each mapping on the domain between 0 and $l-1$. In order to store the set $S = \{x_1, x_2, \ldots, x_n\}$ in the Bloom filter, each element $x_i \in S$ is hash coded using the *k* hash functions and all bits having indices $h_j(x_i)$ for $1 \leq j \leq k$ are set to 1. If a bit was set to 1 before, no change is made.

In general, set membership can be checked by hashing the candidate element *y* using the same *k* hash functions. If all bits having indices $h_i(y)$ in the Bloom filter are already set to 1, *y* is presumably a member of the set *S*. There is a probability that the check indicates membership of *y* in *S* when in fact it is not. It is obvious that the probability of false positive cases depends on the bit array length *l*, the number of hash functions *k*, and the number of elements in *S* denoted by *n*. On the other hand, if at least one of the bits is found to be 0, *y* is definitely not a member of the set *S*.

2.2 Illustrating Example

Suppose the similarity of two surnames should be computed. At first, both surnames are split into sets of *q*-grams. Using 2-grams (usually called bigrams), the 2-gram similarity between the input strings _SMITH_ and _SMYTH_ (padded on both sides with blanks) can be computed with the Dice coefficient as $D_{A,B} = \frac{2 \cdot 4}{(6+6)} = \frac{2}{3}$. If we want to compute the similarity between those strings without revealing the bigrams, we must use an encryption. To accomplish this, we store the *q*-grams of each name in a separate Bloom filter using *k* HMACs. Then we compare the Bloom filters bit by bit and calculate a similarity coefficient.

Figure 2.2-1
Example of the use of two Bloom filters for the privacy-preserving computation of string similarities



In all, 8 identical bit positions are set to 1 in both Bloom filters. In total, 11 bits in A and 10 bits in B are set to 1. Using the Dice coefficient, the similarity of the two Bloom filters is $\frac{2 \cdot 8}{(10+11)} \approx .762$ (plain text bigrams: .66). Therefore, the similarity between two strings can be approximated by using the Bloom filters alone. Using Bloom filters, approximate string matching is possible by a third party *C* despite the privacy of the initial identifiers.

2.3 Simulation study

In order to test the effect of different numbers $(k = 5, 10, 25, 50)$ of hash functions on the performance of Bloom filters, similarities based on Bloom filters with a fixed filter length of 1,000 bits were compared with similarities based on unencrypted 3-grams (trigrams) of simulated data. For the simulation study, 1,000 surnames were sampled from the electronic German phone book. Lines containing just a single character were removed, umlauts and the German "ß" converted and blanks, non alphabetic characters and most common surname components like "von" were deleted. A second list of names to be matched with the original surnames was generated in a copy of the file by changing exactly one character per name with probability $p = .2$ at a randomly chosen position in the name. Therefore two files with 1,000 surnames 20% of which differ at one position were used for computing string similarities.

As is the norm in the information retrieval literature, the criteria of *recall* and *precision* were used to determine the linking effectiveness of the method. For a given level of similarity φ, a pair of records is considered as a match if the pair is actually a true pair, all other pairs are called non-matches. Based on the common classification for true positive (*TP*), false positive (*FP*), false negative (*FN*) and true negative (*TN*) pairs, the comparison criteria are defined as

$$recall = \frac{\sum TP}{\sum TP + \sum FN} \tag{1}$$

$$precision = \frac{\sum TP}{\sum TP + \sum FP} \tag{2}$$

Plotting precision and recall for different similarity values φ as a curve in a precision-recall-plot shows the performance of a string comparison method. A procedure with a better performance will have a curve in the upper right of the plot.

3. Results

Figure 3-1
Comparison of precision and recall for Bloom filters with unencrypted trigrams using simulated data

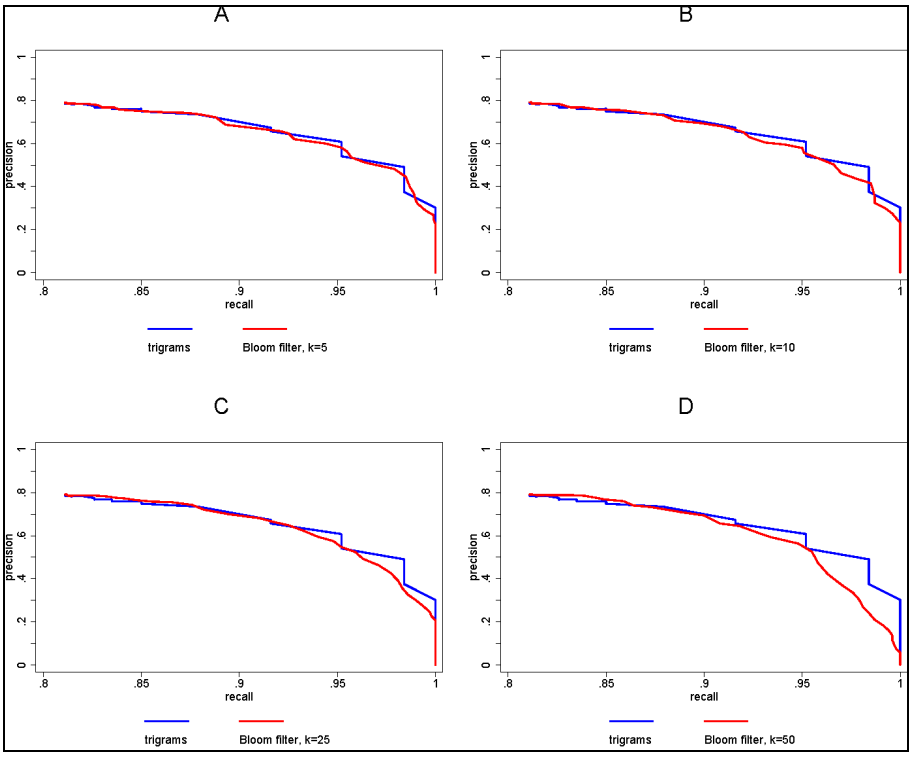


Figure 3-1 shows the results for the first simulation study. The precision versus recall curves in subfigure A are very similar. This is due to the fact that the probability of different trigrams being mapped to the same bit is very low (this probability is given by equation 1), since just five hash functions map the trigrams of a name onto 1,000 bits. When 10 hash functions are being used, the performance of the Bloom filter method is still very similar to the unencrypted trigrams (subfigure B). A small difference between the methods can be seen with 25 hash functions (subfigure C). However, even for 50 hash functions, the difference is not

large (subfigure D). To summarize: Inspection of the precision versus recall plots in figure 2 shows that if the number of hash functions is increased, the difference between the curve of the Bloom filter method and the curve of unencrypted trigrams also increases. However, at least up to 25 hash functions the Bloom filter method performs quite well compared with the unencrypted trigrams.

4. Further Evaluation

In Schnell et al. (2009), we described a full protocol for probabilistic record linkage using Bloom filters in detail. In the context of an evaluation of different probabilistic record linkage procedures for research purposes, we conducted a test of the proposed procedure on two German private administration databases. The performance of the private record linkage protocol using the Bloom filter method for computing string similarities was compared with the performances of unencrypted bigrams and a German phonetic encoding within the same probabilistic record linkage setting. To accomplish the probabilistic record linkage runs, the "Merge ToolBox"[2] was used. The performance of the Bloom filter method was quite comparable to the performance of the unencrypted trigrams and outperformed the phonetic encoding.
Durham et al. (2010) also evaluated the Bloom filter method within probabilistic record linkage. Based on their tests with data containing simulated errors they conclude that the method is applicable with medical data files and that run times are practical for real world use.
In the present proceedings, we also report on a comparison of the Bloom filter method with other approaches to determine the similarity between two encrypted strings (Bachteler et al., 2010). Using experimentally generated human data, the performance of the methods proposed by Pang and Hansen (2004) and Scannapieco et al. (2007) were compared to the Bloom filter method by examining precision-recall plots and run times. The results shows poor precision and recall and long run times by the protocol of Pang and Hansen and good linkage results and a moderate run times by the protocol of Scannapieco et al. However, the Bloom filter method yielded the best linkage results and lowest run times.

5. Conclusions

In this article, we demonstrate a solution for the private record linkage problem with encrypted identifiers allowing for errors in identifiers. The solution is based on similarity computations using Bloom filters with HMACs on *q*-grams. This method has been tested successfully on simulated test data. We were able to show that the linkage results are comparable to non-encrypted identifiers.
Since the method can be easily enhanced and has a low computational burden, the protocol might be useful for many applications requiring privacy-preserving record linkage. The method can be readily incorporated in existing record linkage software by adding a simple subroutine that computes the Dice coefficient from two Bloom filters.

References

Bachteler, T., R. Schnell, and J. Reiher (2010), "An Empirical Comparison of Approaches to Approximate String Matching in Private Record Linkage", *Proceedings of Statistics Canada Symposium 2010. Social Statistics: The Interplay among Censuses, Surveys and Administrative Data*.

Bloom, B. H. (1970), "Space/time Trade-offs in Hash Coding with Allowable Errors", *Communications of the ACM*, 13, pp. 422-426.

Bouzelat, H., C. Quantin, and L. Dusserre (1996), "Extraction and Anonymity Protocol of Medical File", *Proceedings of the 1996 AMIA Annual Fall Symposium*, pp. 323-327.

Camps, R., and J. Daude (2003), "Improving the Efficacy of Approximate Searching by Personal-Name", *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems*, pp. 70-76.

[2] For academic use, the record linkage program Merge ToolBox (MTB) is freely available from http://www.record-linkage.de/. For further information see Schnell et al. (2004).

Churches, T., and P. Christen (2004), “Some Methods for Blindfolded Record Linkage”, *BMC Medical Informatics and Decision Making*, 4.

Durham, E., Y. Xue, M. Kantarcioglu, and B. Malin (2010), “Private Medical Record Linkage with Approximate Matching”, *Proceedings of the 2010 American Medical Informatics Association Annual Symposium*, pp. 182-186.

Herzog, T. N., F. Scheuren, and W. E. Winkler (2007), *Data Quality and Record Linkage Techniques*, New York: Springer.

Hristescu, G., and M. Farach-Colton (1999), “Cluster-preserving Embedding of Proteins“, *DIMACS Technical Report* 99–50, DIMACS Center for Discrete Mathematics & Theoretical Computer Science.

Jain, N., M. Dahlin, and R. Tewari (2005), “Using Bloom Filters to Refine Web Search Results”, *Proceedings of the Eight International Workshop on the Web & Databases*, pp. 25-30.

Pang, C., and D. Hansen (2006), “Improved Record Linkage for Encrypted Identifying Data”, *Proceedings of the 14th Annual Health Informatics Conference*, pp. 164-168.

Schnell, R., T. Bachteler, and S. Bender (2004), “A Toolbox for Record Linkage”, *Austrian Journal of Statistics* 33, pp. 125-133.

Schnell, R., T. Bachteler, and J. Reiher (2009), “Privacy-preserving Record Linkage using Bloom Filters”, *BMC Medical Informatics and Decision Making*, 9.

Scannapieco, M., I. Figotin, E. Bertino, and A. K. Elmagarmid (1996), “Privacy Preserving Schema and Data Matching”, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 653-664.

Trepetin, S. (2008), “Privacy-Preserving String Comparisons in Record Linkage Systems: A Review”, *Information Security Journal*, 17, pp. 253-266.

Verykios, V. S., A. Karakasidis, and V. K. Mitrogiannis (2009), “Privacy Preserving Record Linkage Approaches”, *International Journal of Data Mining, Modelling and Management*, 1, pp. 206-221.