German Record Linkage Center
WORKING PAPER SERIES
NO. WP-GRLC-2011-01 | JULY 1, 2011

A Simplified Variant of a Protocol for Privacy-preserving String Comparison

Tobias Bachteler | Jörg Reiher | Rainer Schnell

A Simplified Variant of a Protocol for Privacy-preserving String Comparison

Tobias Bachteler, Jörg Reiher, and Rainer Schnell

German Record Linkage Center
University of Duisburg-Essen, D-47057 Duisburg, Germany
tobias.bachteler@uni-due.de

1 Introduction

Combining multiple databases with additional information on the same person is increasingly occurring throughout research. Whenever feasible, the databases are merged using a unique identification number. Otherwise, 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, which is problematic because linking encrypted identifiers can result in serious complications. The problem of finding records that represent the same individual in separate databases without revealing the identity of the individuals is called the "privacy-preserving record linkage" problem [1].

Due to the frequency of spelling and typographical errors, in practical applications record linkage algorithms have to use string similarity functions. 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, algorithms for computing string similarity functions with encrypted identifiers are essential for approximate string matching in private record linkage.

The problem of how to accomplish approximate string comparison in private record linkage can be compactly stated as follows: Consider two database owners A and B holding lists of strings (e. g. names) $S_a = \{a_1, \ldots, a_n\}$ and $S_b = \{b_1, \ldots, b_m\}$. Compute string similarities of all pairs $(a, b) \in A \times B$ such that all the names remain private to the database holders.

[2] proposed a protocol for determining string similarities on encrypted strings based on an embedding of the strings in a multi-dimensional space. For reasons of efficiency, they adopted *SparseMap* as embedding method. The heuristics of *SparseMap* are intended to improve the efficiency of the embedding process in exchange for some accuracy of similarity searches in the set of embedded strings. We suspected the protocol performing better if we omit *SparseMap* from the protocol.

2 A Protocol for Privacy-preserving Record Linkage

The the protocol described by [2] is based on an ingenious idea to solve the problem of private approximate string comparison. The idea is to embed the strings in an Euclidean space and then to compare their embeddings instead of their plain text values. Specifically, two data holders, holding lists of names, build an embedding space from random strings and embed their respective strings therein using the *SparseMap* method ([3,4]). Then, each data holder sends the embedded strings to a third party which 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. Using *SparseMap* allows the mapping of strings into the vector space avoiding prohibitive computational costs. This is accomplished by the reduction of dimensions using two heuristics, *greedy resampling* and *distance approximation*. However, the experiments in [2] indicate that the linkage quality is significantly affected by applying the *greedy resampling* heuristic.

3 Description of the Protocol Variant

The deteriorating effects of the *greedy resampling* heuristic shown in [2] lead directly to the idea to use a variant of the protocol which skips both *SparseMap* heuristics completely. The heuristics are intended to improve the efficiency of the embedding process in exchange for accuracy of similarity searches in the set of embedded strings. We assumed that if we fill the reference sets with just a few reference strings, we can balance the loss of efficiency involved when omitting *distance approximation* as well.

In our variant, the data holders agree on a number k of reference sets and fill each reference set $A_i \in R = \{A_1, A_2, \ldots, A_k\}$ with a fixed number n of random strings of a fixed length L. The k reference sets are used to embed the names in the k-dimensional space. The coordinates for a name are given by the distances between the name to the closest random string in each of the k reference sets in terms of the edit distance, $\delta(x, y)$. Note that no distance approximation is used here. That is, the ith Component of the embedding vector is given by $\min_{y \in A_i} \{\delta(x, y)\}$.

Protocol:

1. A and B agree on a number n of reference sets, a string length L, and a edit distance variant.
2. A creates k reference sets, each consisting of n random strings of length L.
3. A embeds each name from S_a by calculating the minimal distance δ_{min} of the name with the n random strings in each of the k reference sets and stores the resulting embedding vector V_a along with a ID-number ID_a in a list $A.enc$.
4. A transmits $A.enc$ to C.
5. A transmits the k reference sets to B.
6. B embeds each name from S_b by calculating the minimal distance δ of the name with the n random strings in each of the k reference sets and stores the resulting embedding vector V_b along with a ID-number ID_b in a list $B.enc$.
7. B transmits $B.enc$ to C.
8. For each possible pair of names from $A.enc$ und $B.enc$, C calculates the euclidean distance δ^E and stores as the result a list consisting of tuples ID_a, ID_b, δ^E.

The security of this protocol variant corresponds to the original version since no additional data is transmitted. Giving up the *SparseMap* heuristics simplifies the protocol considerably and was expected to improve the linkage quality. On the other hand, a certain loss of efficiency was to be expected subject to the number of strings in the reference sets.

4 Empirical Evaluation

4.1 Test data

For empirical testing we used experimentally generated human data. 1,500 surnames were randomly selected from a town register. Then, 15 audio tapes with 100 different names each were recorded with a female voice. Each tape was played to one of 15 groups of students with an average of 19 students per group. Each student wrote the name heard, 13 groups by hand, two groups typed the names. Finally, all student notes were typed by a single typist resulting in 1,286 unique original names (file A) and 27,296 potentially erroneous names (file B). We determined the similarity of each pair of names $(a, b) \in A \times B$ and determined linkage quality by examining precision-recall plots. As an external standard of comparison we determined the Damerau-Levenshtein distance from the plain text names.

4.2 Precision-Recall Plots

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.

4.3 Linkage quality: Evaluation of the new variant

To evaluate the new, simplified variant of the protocol, we varied three parameters: The string length L, the size of the reference sets n, and the number of reference sets or dimensions of the embedding k. We began with n fixed at 10 and k fixed at 50 and varied L. Because there is randomness built in the protocol via the random strings in the reference sets, we replicated each individual test ten times. The precision-recall curves show the of average recall at 10 fixed precision values. Minimum and maximum recall are marked by caps at each precision value.

Fig. 1: **New variant: Precision-recall curves using $k = 50$ dimensions, $n = 10$ strings in each reference set and various string lengths**

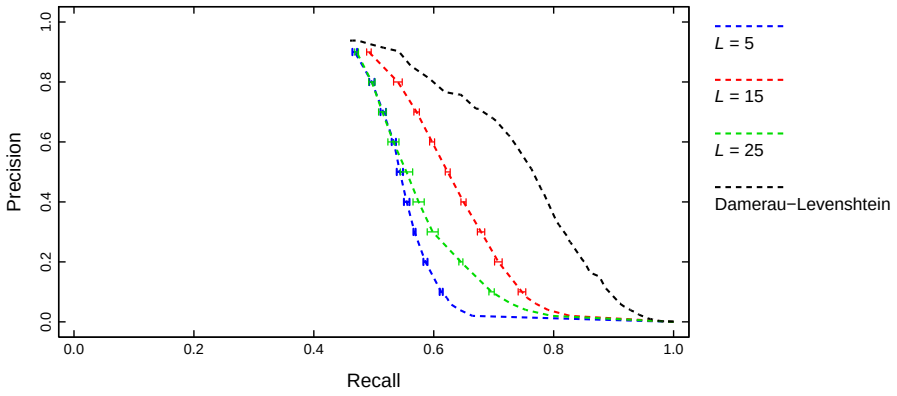
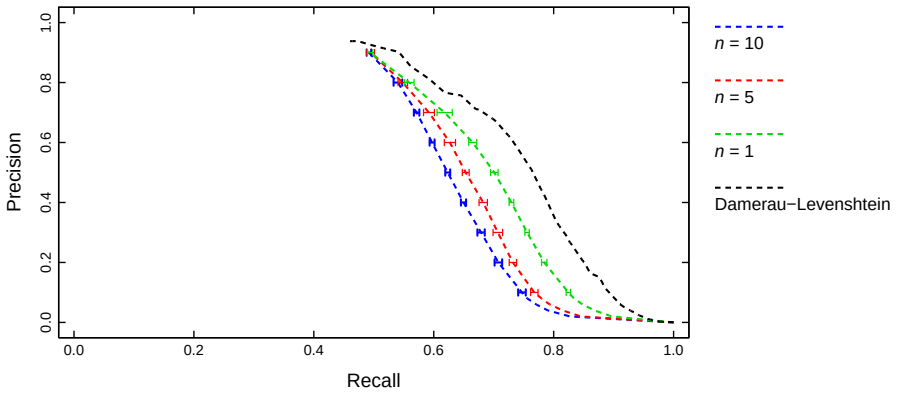


Figure 1 shows precision and recall depending for string lengths 5, 15, and 25. The setting with a string length of 15 clearly outperforms the other settings, so we fixed L at 15 and varied n, the number of strings in each reference set.

Fig. 2: **New variant: Precision-recall curves using $k = 50$ dimensions, string length $L = 15$ and various numbers of strings in each reference set**



In Figure 2 the results for three values of n are displayed. The finding is that the fewer strings in the reference sets, the better the linkage quality. The setting with $n = 1$ gives the best result in terms of precision and recall.

Fig. 3: **New variant: Precision-recall curves using string length $L = 15$, $n = 1$ string in each reference set and various numbers of dimensions k**

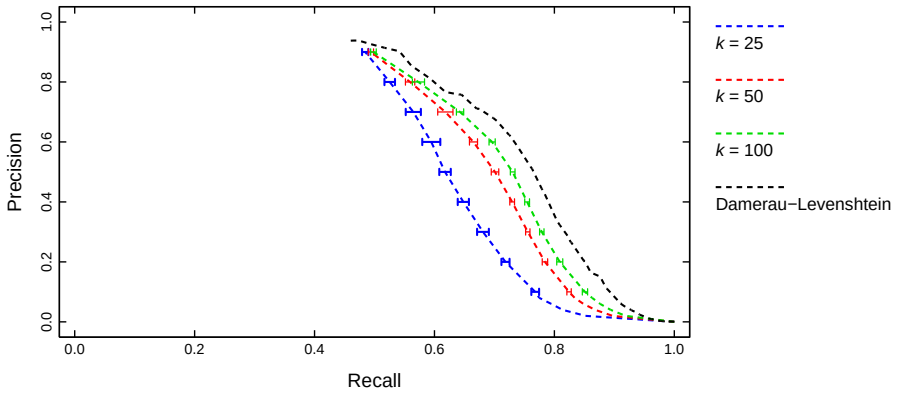


Figure 3 2 shows the precision-recall curves for fixed L and n 25, 50, and 100 reference sets or dimensions. Using more dimensions leads to improved performances of the protocol. However, the additional gain of changing from 50 to 100 dimensions is smaller than the gain of changing from 25 to 50, indicating diminishing returns of increasing the number of dimensions. As the best performing parameter combination we identified a string length of $L = 15$ along with $n = 1$ string in each reference set and a dimensionality of $k = 100$.

4.4 Linkage quality: Comparison of original protocol and the new variant

In [5], we compared different parameter settings of the original protocol using the data described in section 4.1. In our tests we found a string length of 20 optimal. Figure 4 shows the precision-recall curves for different numbers of dimensions. Taking runtime into consideration, we found $k = 64$ reference sets or dimensions performing best.

Fig. 4: **Original protocol: Precision-recall curves using string length $l = 20$ and various numbers of dimensions k**

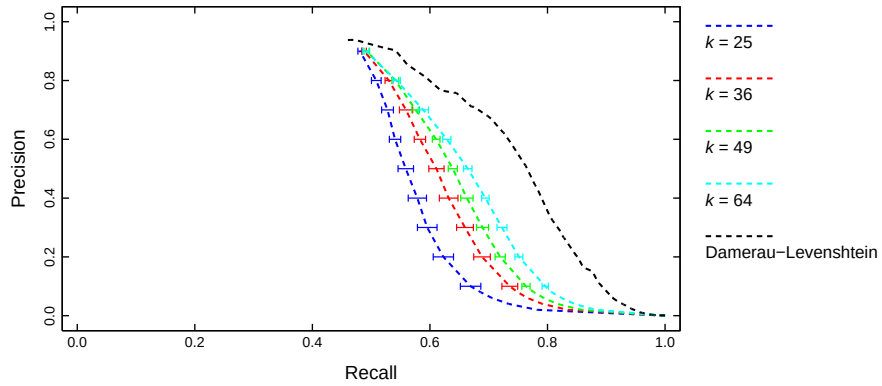


Figure 5 compares the original protocol with the new variant. With the parameters we found to be performing best ($k = 100$, $L = 15$, $n = 1$), the new variant outperforms the original protocol. The distance from the reference line indicates quite a good approximation of the plain text comparison of the names.

However, in this comparison the original protocol is handicapped by a lower number of dimensions used. Thus we conducted another test using $k = 100$ dimensions in the original protocol as well. In figure 6 the resulting precision-recall curve is displayed along with the best setting of the new variant. This time the original protocol performs much better indicating that the number of dimensions is a crucial factor for linkage quality of the protocol. However, the new variant still outperforms the original variant.

Fig. 5: **Comparison of precision and recall of the original protocol ($k = 64$, $l = 20$) and the new variant ($k = 100$, $L = 15$, $n = 1$)**

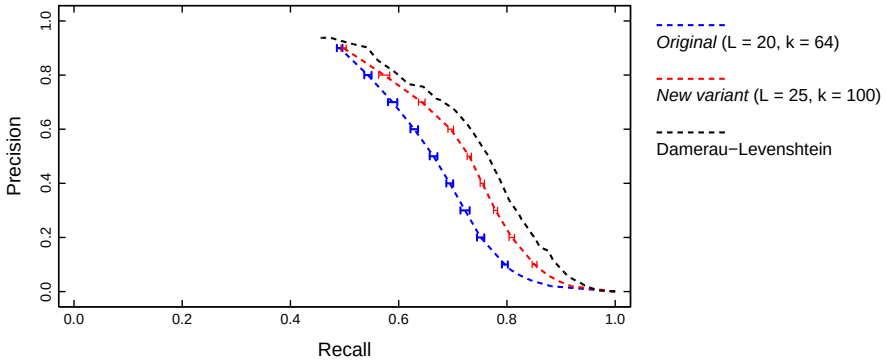
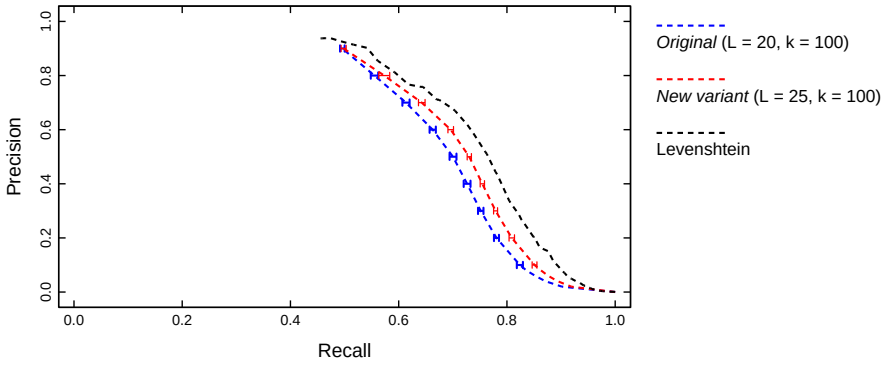


Fig. 6: **Comparison of precision and recall of the original protocol ($k = 100$, $l = 20$) and the new variant ($k = 100$, $L = 15$, $n = 1$)**



4.5 Efficiency

The runtimes of the protocol variants compared in section 4.4 are given by table 1. The original protocol with $k = 64$ required 84:50 minutes both to encrypt and compare the names. The original protocol with $k = 100$ needed almost 7 hours to finish, whereas the new protocol variant shows a runtime comparable of that of the original protocol with $k = 100$.

Table 1: *Comparison of run times (means, minutes)*

Method	*Encryption*	*Matching*	*Total*
Original, $k = 64$	19:25	65:25	84:50
Original, $k = 100$	333:00	86:18	419:18
New variant, $k = 100$	2:54	84:48	87:42

5 Conclusions

In this paper, we report the results of an empirical evaluation of a simplified variant of a protocol for privacy-preserving string comparison proposed in [2]. We supposed that a variant of the protocol which omits the two *SparseMap* heuristics should perform better and that a lower number of random strings in the reference sets would balance the efficiency advantages of *SparseMap* in this instance.

In our empirical tests, we found the simplified protocol variant performing better than the original variant. With $k = 64$ dimensions, the original protocol is slightly more efficient but considerably less effective in terms of linkage quality. With $k = 100$ dimensions the original protocol comes closer but is still less effective than the simplified variant. However, with $k = 100$ dimensions the original variant is far less efficient than the new variant.

Thus, the empirical findings confirm our supposition that it would be possible to balance the efficiency advantages of *SparseMap* while getting better linkage results by omitting the two *SparseMap* heuristics in a simplified protocol variant.

References

1. Rob Hall and Stephen E. Fienberg. Privacy-preserving record linkage. In Josep Domingo-Ferrer and Emmanouil Magkos, editors, *Proceedings of the 2010 international conference on Privacy in statistical databases*, PSD'10, pages 269–283, Berlin, Heidelberg, 2010. Springer-Verlag.
2. Monica Scannapieco, Ilya Figotin, Elisa Bertino, and Ahmed Elmagarmid. Privacy preserving schema and data matching. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data: 12–14 June 2007; Beijing, China*, pages 653–664, New York, 2009.04.10 2007. ACM.
3. G Hristescu and M Farach-Colton. Cluster-preserving embedding of proteins. *DIMACS Technical Report 99-50, DIMACS Center for Discrete Mathematics & Theoretical Computer Science*, 1999.
4. Gisli R. Hjaltason and Hanan Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.
5. Tobias Bachteler, Rainer Schnell, and Jörg Reiher. An empirical comparison of approaches to approximate string matching in private record linkage. In *Proceedings of Statistics Canada Symposium 2010. Social Statistics: The Interplay among Censuses, Surveys and Administrative Data (forthcoming)*, 2010.

IMPRINT

Publisher

German Record-Linkage Center
Regensburger Str. 104
D-90478 Nuremberg

Template layout

Christine Weidmann

www.record-linkage.de