Private Record Linkage: Comparison of Selected Techniques for Name Matching

Pawel Grzebala(✉) and Michelle Cheatham

DaSe Lab, Wright State University, Dayton, OH 45435, USA
{grzebala.2,michelle.cheatham}@wright.edu

Abstract. The rise of Big Data Analytics has shown the utility of analyzing all aspects of a problem by bringing together disparate data sets. Efficient and accurate private record linkage algorithms are necessary to achieve this. However, records are often linked based on personally identifiable information, and protecting the privacy of individuals is critical. This paper contributes to this field by studying an important component of the private record linkage problem: linking based on names while keeping those names encrypted, both on disk and in memory. We explore the applicability, accuracy and speed of three different primary approaches to this problem (along with several variations) and compare the results to common name-matching metrics on unprotected data. While these approaches are not new, this paper provides a thorough analysis on a range of datasets containing systematically introduced flaws common to name-based data entry, such as typographical errors, optical character recognition errors, and phonetic errors.

1 Introduction and Motivation

Data silos, in which organizations keep their data tightly isolated from other systems, are a major barrier to the effective use of data analytics in many fields. Unfortunately, when the data in question involves information about people, integrating it often necessitates querying or joining based on personally identifiable information (PII) that could be used to explicitly identify an individual. As recent security breaches at organizations ranging from Target to the United States Postal Service have made clear, it is important to protect PII, both while it is at rest on a system and when it is read into memory. The goal of this effort is to explore the applicability, accuracy, and speed of existing algorithms for querying and joining databases while keeping the PII within those databases protected.

This work focuses particularly on the situation in which a data provider maintains a database which authorized subscribers are able to query. For instance, consider a company that maintains a database containing its customer data. The company wishes to allow third party entities who have contracted with it to access the information in this database.[1] At the same time, the company wants

[1] A standard access control system to allow authorized consumers to query the database while preventing unauthorized users from doing so is assumed to be in place.

H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 593–606, 2016.
DOI: 10.1007/978-3-319-34129-3_36

to limit its vulnerability to data breaches by keeping the data encrypted as much as possible, including while it is stored in the database and when it is loaded into memory to do query processing. For instance, if an attacker gets access to the system on which the database resides, he should not be able to see the raw data values, either on disk or in memory.

Even though this situation occurs frequently, research on private record linkage tends to focus more on a different use case, in which the data provider and the data consumer do not fully trust one another. This typically leads to solutions involving trusted third parties or asymmetric cryptography that go beyond the requirements of this ubiquitous application scenario, and these additional, unneeded capabilities negatively impact performance. For instance, because access control mechanisms are already in place, this work is not concerned about the data consumer (who has paid to access the information in the database) gaining knowledge of any, or even all, of the records in the database. Furthermore, this project is not concerned about the data consumer preventing the data provider from gaining knowledge about what queries are being made. Rather, the present use case allows a system in which the data consumer submits a query containing the raw PII values, these values are encrypted using symmetric key cryptography[2], and the encrypted values are then used to query the database.

This work focuses on supporting privacy-preserving querying and merging on string attributes and does not consider numeric data. While private record linkage based on numeric fields of is course an important capability to establish, the techniques involved for this are distinctly different than for string-based linking. Furthermore, string attributes, in particular person names, are a particularly common linkage point between datasets. We therefore leave the challenge of numeric attributes for future work and focus on name-based linking here. The requirements of our target application scenario require PRL methods that support encryption and do not need to act directly on the raw field values, so approaches that utilize the original string values at any stage in the process are not suitable in this case. Because names are frequently misspelled, mispronounced, or mistyped, it is important for the approach to support fuzzy (approximate) matching as well as exact matching. This fuzzy matching should be particularly tailored to support the types of lexical variations specific to names. No data should be decrypted, even in memory, until a match is ensured. In this paper we analyze the accuracy and efficiency of several metrics that meet these requirements and compare those results to that of standard name-matching methods employed on unencrypted data. The paper focuses entirely on technical considerations of the targeted use case. Laws and regulations also have a bearing on this application, but that aspect is not addressed here due to wide variance between legal jurisdictions and the authors' lack of legal expertise.

Note that nothing in this application scenario places any restrictions upon the infrastructure in which the data records are stored. In particular, the results presented here can be applied directly, with no modification, to data stored as

[2] Note that this exposes the raw PII values in memory, though only those in the query, not those in every database record.

RDF triples in accordance with the linked data principles. This work therefore joins a growing body of literature regarding how linked data can be secured while retaining its utility for those authorized to access it [5,7].

The main contributions of this paper are:

- The usage of an advanced name matching benchmark generation tool to analyze the performance of several different name-based similarity metrics in a nuanced way. In our analysis we consider numerous realistic sources of errors and study the effect of the threshold value applied to each of the metrics.
- The accuracy of the privacy-preserving similarity metrics is compared to that of standard string metrics on unprotected data in order to establish the accuracy lost in support of data privacy.
- The computational efficiency of the privacy-preserving similarity metrics is also compared to that of standard string metrics.

The rest of this paper is organized as follows: In Sect. 2 we provide an overview of some related work and briefly discuss the challenges that make record linkage on names difficult. Section 3 introduces the metrics and algorithms used to perform record linkage in this study. This includes the string similarity metrics for unencrypted data which are used as a baseline for comparison purposes and the metrics relevant to private record linkage. Section 4 analyzes and evaluates the performance of the algorithms mentioned in Sect. 3 in terms of accuracy and computational efficiency. Finally, Sect. 5 concludes the paper by summarizing the results and provides an outlook to future work.

2 Background

There have been numerous approaches to solving the problem of record linkage based on person names. A comprehensive overview of several name matching techniques was provided by Snae in [9]. Snae describes four different types of name matching algorithms and compares them in terms of accuracy and execution time: spelling analysis based algorithms (Guth and Levenshtein), phonetic based algorithms (Soundex, Metaphonez, and Phonex), composite algorithms (combination of sound and spelling based methods, e.g. ISG), and hybrid algorithms (combination of phonetic and spelling based approaches, e.g. LIG). The hybrid algorithms were recommended for many name based record linkage applications because of their flexibility that allows them to be easily tuned for specific use cases. However, the results indicated that there is no single best method for name matching. In the conclusion, the author suggests that the choice of the name matching algorithm should depend on the specific application needs. Moreover, this work doesn't take into consideration the important aspect of our study, which is linking records while keeping them encrypted.

As mentioned previously, many existing techniques for private record linkage assume that the two parties involved do not want to reveal their data to the other party. One way this is commonly achieved is by developing algorithms

that avoid directly comparing the records to be linked. For example, in the two-party protocol presented by Vatsalan and his colleagues in [11], two database owners compute similarity values between the records in their dataset and public reference values. Then, the similarity values are binned into intervals and the bins are exchanged between the two database owners. Based on the exchanged bins the protocol uses the reverse triangular inequality of a distance metric to compute the similarity values of two records without revealing the records themselves. Another, somewhat similar, two-party protocol was proposed by Yakout et al. in [12]. In this approach, each database owner converts all of their records into vector representations that are later mapped to points in a complex plane. The planes are then exchanged between the owners in order to identify pairs of points that are in proximity of each other. To calculate similarity values between the candidate vectors, the Euclidean distance of two records is computed using a secure distance computation. These two approaches are typical of many existing PRL techniques and, like the majority of those techniques, they implicitly assume that the records to be linked are not encrypted. We now turn our attention to examples of the few approaches that do not make this assumption.

One important thing to note is that not all string similarity metrics can be applied to the problem of name-based private record linkage. In order for a metric to be usable in this scenario, the metric must not require access to individual characters within this string. This is because any such metric would have to "encrypt" a string character-by-character, which is essentially a classical substitution cipher that is not at all secure. This eliminates common metrics such as Levenshtein and Monge Elkan from consideration.

Among the techniques that support approximate matching for linking records are the Soundex and q-gram string similarity metrics. The Soundex metric was originally designed as a phonetic encoding algorithm for indexing names by sound. [1] Soundex encodes a string representing a name into a code that consists of the first letter of the name followed by three digits, by applying a set of transformation rules to the original name. When two Soundex encodings are compared, the comparison is an exact match rather than approximate comparison but common name mispronunciations will not cause the algorithm to miss a match[3]. To use Soundex for private record linkage, both the name and the phonetic encoding are stored in the database in encrypted form for each record, but the encrypted phonetic encoding is the one used to respond to queries. The comparison is still an exact rather than fuzzy comparison, but because it is now being done on a phonetic encoding, common misspellings or other slight

[3] In 1990 Lawrence Philips created a phoenetic algorithm called Metaphone that improves upon Soundex by considering numerous situations in which the pronunication of English words differs from what would be anticipated based on their spelling [8]. Metaphone was not considered for this effort because the extensions that it makes beyond Soundex are primarily intended to improve the performance on regular words rather than on names; however, the metric does fit the requirements for use in this application, and will be considered during our future work on this topic.

differences will not cause the algorithm to miss matching records. This was the approach suggested in [3].

Another of the string similarity metrics that *can* be used is q-grams. A q-gram is created by splitting a string into a set of substrings of length q. An example of a q-gram, given $q = 2$ and the input string *Alice*, is {"*Al*", "*li*", "*ic*", "*ce*"}. As with the Soundex approach, in order to use q-grams for name-based private record linkage additional information must be stored with each record. In the case of *q-grams*, the person's name is divided into q-grams, each of the substrings in the set of q-grams is encrypted, and those encrypted substrings are also stored as part of the record. The amount of similarity between two records is then computed as the degree of overlap between these set of encrypted q-grams for each record. Each individual substring is compared based on exact match. The degree of overlap is computed using a traditional set similarity metric such as Jaccard or Dice, which are calculated as follows:

$$\text{Jaccard} = \frac{\mathsf{grams}_{\text{common}}}{\mathsf{grams}_1 + \mathsf{grams}_2 - \mathsf{grams}_{\text{common}}}$$

$$\text{Dice} = \frac{2 \times \mathsf{grams}_{\text{common}}}{\mathsf{grams}_1 + \mathsf{grams}_2},$$

where $\text{grams}_{\text{common}}$ corresponds to the number of q-grams that are common to both strings, grams_1 to the number of q-grams in the first string, and grams_2 to the number of q-grams in the second string. The intuition behind using q-grams to compare two names is that a typo, misspelling, or other variation will only impact a limited number of substrings and therefore similar strings will still have a high degree of overlap and thus a high similarity value. The downside is that the order of the substrings is not considered, so it is possible for two very different strings, such as "stop" and "post" to have very high similarity according to this metric.

A more in-depth review of techniques proposed to achieve private record linkage can be found in [2].

In this work, we have evaluated the performance of the Soundex and q-gram algorithms for name-based private record linkage in the scenario described in the introduction. Because it is unrealistic to expect a privacy-preserving record linkage algorithm to perform better than a linkage method that does not provide any protection for the data, we have compared the performance of Soundex and q-gram to the performance of some traditional string similarity metrics on unencrypted data. Specifically, we have used Levenshtein and Jaro-Winkler, two of the most commonly used string similarity metrics, as a baseline. Levenshtein is an edit distance metric. It simply counts the number of edits (insertions, deletions, or substitutions) that must be applied to one string in order to transform it into another one. For example, the Levenshtein distance between "Michelle" and "Micheal" is 2. Jaro-Winkler is based on the Jaro metric, which counts the number of "common" characters of two strings. Characters are considered common when the difference between their indexes is no greater than half of the length of the longer string. Jaro also takes into consideration the number of character transpositions. The Jaro-Winkler version of the algorithm increases

the similarity value returned by Jaro if the two strings begin with the same sequence of characters and differences appear only in the middle or at the end of string [2].

Name matching has been researched for many years and numerous studies have proven that it is not an easy task. This is because a name's spelling can be malformed in a wide variety of ways, including punctuation, abbreviation, pronunciation, spelling, the order of writing, use of prefixes, typos, or optical recognition errors to name a few. In addition, privacy concerns have made it very difficult to find publicly available data that can be used for benchmark purposes, particularly a collection of names that accurately reflects worldwide name distribution rather than being US-centric. This lack of suitable benchmarks was a considerable challenge during this study, leading to the use of a newly-available name matching benchmark generation system, described in Sect. 4.

3 Approach

We analyzed the performance of several string similarity metrics for linking encrypted records. The metrics considered were Soundex and several variations of the q-gram technique. This performance was compared against those of Jaro and a normalized version of Levenshtein on the unencrypted data. The data and Java source code are available from https://github.com/prl-dase-wsu/prl-technique-comparison.

We used two metrics based on q-grams. The first is q-grams with $q = 2$ (also called bigrams). Because studies have shown [6] that padding the input string with a special character (one that never appears as part of any string in the dataset) at the beginning and the end of string can increase the accuracy when comparing two different q-grams we also tried padded q-grams with $q = 2$. Both q-grams and padded q-grams were compared using two different similarity coefficient methods, Jaccard and Dice.

The string metrics that were used on unencrypted data, Jaro and Levestein, were introduced in Sect. 2. To formally define both algorithms, the similarity value of two strings returned by the Jaro algorithm is calculated as follows:

$$\text{Jaro} = \frac{\frac{c}{s_1} + \frac{c}{s_2} + \frac{c - t}{c}}{3},$$

where c is the number common characters in both strings, s_1 is the length of the first string, s_2 is the length of the second string, and t is the number of transpositions (the number of common characters that are not in sequence order, divided by 2). Since all of the metrics used in this study return a value between 0.0 (when strings are completely different) and 1.0 (when string are the same) we modified the original Levenshtein algorithm so that it returns a similarity value that falls in the same range. The Normalized Levenshtein formula is defined as follows:

$$\text{NormalizedLevenshtein} = 1 - \frac{\text{Levenshtein}}{\max(s_1, s_2)},$$

where *Levenshtein* is the number of replacements needed to transform the first string into the second string, s_1 is the length of the first string, and s_2 is the length of the second string.

The benchmark datasets used in this study were created by using the advanced personal data generation tool called "GeCo" and developed by K-N. Tran et al. [10] The tool was created to address the issue of lack of publicly available data that contains PII information. GeCo has two main functionalities: data generation and data corruption. The data generation module provides the user with an interface capable of producing records with five different attribute generation mechanisms. The first two can be used to generate individual attributes such as credit card number, social security number, name, age, etc. The attribute values are created by either user-defined functions or based on frequency look-up files that specify the set of all possible values of an attribute and their relative frequencies. The other three types of attribute generation mechanisms allow the user to produce compound attributes where the attributes' values depend on each other. For example, a compound attribute with fields such as: city, gender, and blood pressure can be created, where the value of the blood pressure depends on the previously generated city and gender values. The second module of GeCo provides users with a sophisticated interface allowing them to corrupt the generated data using six different corruption techniques that simulate real-world errors that can occur during data processing. Those techniques include introducing: (1) missing values (one of the record's fields gets lost), (2) character edits (a random character of a string attribute is inserted, deleted, substituted, or transposed), (3) keyboard edits (simulates a human mistake during typing), (4) optical character recognition (OCR) errors (simulates OCR software mistakes), (5) phonetic edits (replaces substrings with their corresponding phonetic variations), and (6) categorical value swapping (replaces an attribute value with one of its possible variations). The user can also specify numerous other parameters such as: the number of records to corrupt, the number of corruptions applied to a record or single attribute, or the probability of corruption of a particular attribute.

For benchmark purposes we generated a dataset of 10,000 records where each of the records had the following attributes: first name, last name and credit card number. Then, we used the GeCo tool to introduce various types of realistic corruption to the generated dataset. The corrupted datasets produced by the GeCo tool were categorized using three parameters: type of applied corruption technique (Character Edit, Keyboard Edit, OCR Edit, Phonetic Edit, or mix of all), the percentage of original record corruption (high - 10 %, medium - 5 %. or low - 2 %), and the number of corruptions applied to either the first name, last name, or both (1 or 2). This resulted in 30 variations of the dataset. Once the datasets were corrupted we added additional attributes to each of the records from all datasets to be able to perform record linkage using encrypted q-grams and Soundex encodings. Each q-gram array and Soundex encoding were encrypted using 256-bit AES password-based encryption.

To evaluate the performance of the string metrics, the uncorrupted dataset was cross joined with each of the corrupted datasets using each of the string metrics discussed in the previous section. During the join operation only the pairs of records with the highest similarity score that exceeded a threshold value were joined. If the an individual pair of joined records corresponded to the same individual we counted it as a "Correct" join, otherwise the join was counted as "False Positive". If none of the scores returned by a string metric exceeded a threshold value we incremented the "False Negative" count by 1 to indicate that a corresponding record was not found in the other dataset. In a special case, when more than one pair of records had the same highest score, the pair of records that corresponded to the same individual was marked as "Correct" and the rest of the pairs were counted as "False Positive".

4 Evaluation and Analysis

Performing record linkage using each of the seven string metrics (q-grams compared using Jaccard coefficient, q-grams compared using Dice coefficient, padded q-grams compared using Jaccard coefficient, padded q-grams compared using Dice coefficient, Soundex, Levenshtein, and Jaro) between the uncorrupted dataset and the 30 corrupted datasets resulted in a massive amount of statistical data. Instead of presenting the outcome of every single cross join operation, this section summarizes our key findings, with an emphasis on practical advice related to selecting a metric, setting the threshold, and conveying the type of performance that can be expected by someone attempting to do name-based private record linkage.

4.1 Observation 1: Soundex is Not Viable, but (Padded) q-grams are

Figure 1 Shows the results of all of the string metrics on a version of the data in which 10 % of the names have had one (the solid lines) or two (the dotten lines) characters edited. In the single edit case, all versions of the q-gram metric are able to achieve the same, nearly perfect, accuracy on the encrypted data that Levenshtein and Jaro achieve on the encrypted data. The performance of all metrics is lower for the two character edit case, with a top accuracy of 90 % rather than the completely accurate results possible in the single edit situation. However, we again see that the performance of at least the padded versions of the q-gram approach on the ciphertext can match that of Levenshtein and Jaro on the plaintext.

The results for the Soundex metric are not included in Fig. 1 because the results showed that comparing names based on encrypted Soundex encodings is not viable in most of the cases as a record linkage technique. The only exception was noted when the datasets containing records with the phonetic type of record corruption were joined. Still, in the best case scenario only 60.71 % of corrupted data was successfully matched using this technique. Table 1 presents the accuracy of record linkage on all types of corrupted datasets using the Soundex technique.

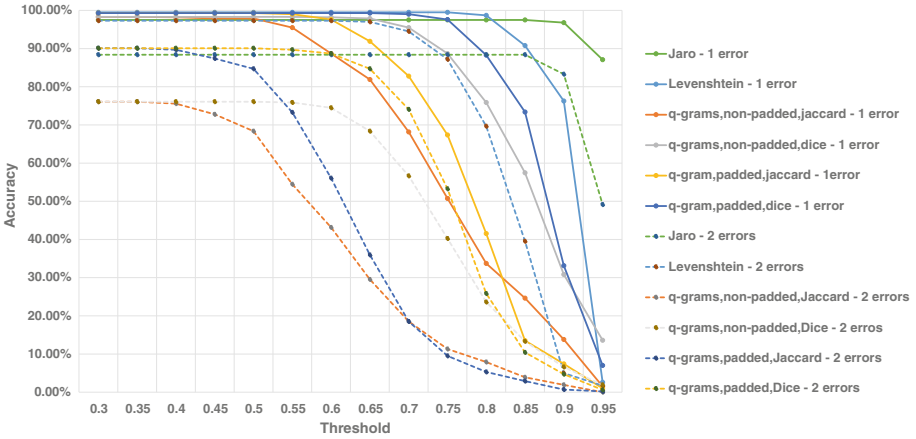

Fig. 1. Illustration of the decrease in accuracy of record linkage of selected string metrics. Solid lines correspond to accuracy when linkage was performed on datasets corrupted with 1 Character Edit, dotted lines with 2 Character Edits.

Table 1. Performance of record linkage based on encrypted Soundex encodings. The percentage values reflect the number of corrupted records that were successfully matched with their uncorrupted versions.

Corruption type	Number of corruptions per record	
	1	2
Character Edit	47.24 %	24.06 %
Keyboard Edit	48.06 %	21.94 %
OCR Edit	38.29 %	13.71 %
Phonetic Edit	60.71 %	43.88 %
Mix	50.82 %	25.47 %

4.2 Observation 2: Dice is Preferable to Jaccard for Calculating *q*-gram Similarity

Out of the four similarity metrics based on q-grams, the ones using the Dice coefficient to measure the similarity between the sets of encrypted q-grams were more accurate. This was the case with q-grams as well as padded q-grams. This can be explained by the fact that Dice favors the occurrences of common q-grams more than Jaccard. As a result, a pair of similar records is likely to have a higher similarity score when calculated using Dice coefficient. To illustrate this, in Table 2 we provide a sample results from record linkage performed against a dataset with the phonetic type of corruption, where 10 % of original records had two phonetic errors introduced. Similar results were recorded for datasets with other types of corruptions.

Table 2. Sample results of record linkage performed against phonetically corrupted dataset showing the performance of q-grams based string similarity metrics

Threshold	Unpadded q-grams						Padded q-grams					
	Jaccard			Dice			Jaccard			Dice		
	Correct	FP	FN	Correct	FP	FN	Correct	FP	FN	Correct	FP	FN
0.30	9837	60	163	9837	60	163	9949	45	51	9949	45	51
0.35	9837	60	163	9837	60	163	9949	45	51	9949	45	51
0.40	9834	57	166	9837	60	163	9948	45	52	9949	45	51
0.45	9814	56	186	9837	60	163	9937	43	63	9949	45	51
0.50	9778	48	222	9837	60	163	9910	38	90	9949	45	51
0.55	9614	29	386	9834	57	166	9790	27	210	9949	45	51
0.60	9506	25	494	9826	57	174	9589	18	411	9946	45	54
0.65	9329	21	671	9778	48	222	9337	18	661	9910	38	90
0.70	9198	18	802	9637	32	363	9170	18	830	9792	27	208
0.75	9101	18	899	9467	22	532	9081	17	919	9538	18	462
0.80	9046	18	954	9264	19	736	9032	17	968	9233	18	767
0.85	9023	18	977	9125	18	875	9011	17	989	9097	17	903
0.90	9009	18	991	9036	18	964	9002	17	998	9021	17	979
0.95	9002	18	998	9009	18	991	9000	17	1000	9002	17	998

4.3 Observation 3: Lower Thresholds are Better for q-grams

Figure 1 illustrates that the threshold value for the Levenshtein and Jaro metrics can be set relatively high without sacrificing accuracy when linking the unencrypted data, which was not the case when the q-gram techniques were used to link the encrypted data. For instance, to achieve an accuracy of 99.5 % when performing linkage against datasets where records contain one corruption of any type, the threshold value applied to the Jaro or Levenshtein metric was set to 0.8 whereas the threshold value applied to q-grams based metrics needs to be set to a value between 0.55 and 0.75 to achieve the same result, depending on the type of corruption applied to the datasets.

Table 2 makes the point that the padded versions of the q-gram metric in particular have better performance when the threshold value is kept low, which as explained in the previous paragraph is the optimal approach. For threshold values up to 0.7 for the Jaccard coefficient and 0.8 for the Dice coefficient, padding the q-grams produces better results. For higher threshold values, the unpadded version is slightly better. The reason behind this is that similarity scores calculated using padded q-grams are higher when the differences between the strings used to generate the q-grams appear in the middle of the strings. [2] When the differences appear at the beginning or at the end of strings the similarity scores are lower because the number of common q-grams is smaller. Statistically, the differences between strings appear more often in the middle, which explains why the padded q-grams can produce higher similarity scores for the majority of corrupted data. This pattern occurred in all of the results produced during this study.

4.4 Observation 4: Some Types of Errors are Worse than Others

Out of all corrupted datasets the worst performance in terms of accuracy and the number of false positives found was "achieved" when the datasets with OCR Edits were linked. This is most likely due to the fact that some of the mistakes that OCR Edits introduce are replacements of two characters in place of one character, or vice versa. For instance, character "m" can be replaced with "rn" and the string "cl" can be replaced by the character "d". Those kind of replacements can have a significant negative impact on the string similarity scores produced by all of the metrics. The best performance results were recorded when the datasets corrupted with Character Edits were linked, those are presented in Fig. 1. Figure 2 illustrates the accuracies of linking datasets corrupted with OCR Edits errors. The accuracies of datasets corrupted with Keyboard Edits, Phonetic Edits, and a mix of all types of edits fall in between the accuracies presented in Figs. 1 and 2.

Another pattern common for the results obtained from linking all types of corrupted datasets was a significant drop in accuracy when the corrupted records contained more than one error of any type. For instance, for $Threshold = 0.85$ the accuracies of the Jaro, Levenshtein, unpadded q-grams compared using the Dice coefficient, and padded q-grams compared using the Dice coefficient were 97.5 %, 90.79 %, 57.46 %, and 73.37 % respectively when there was only one error of the Character Edit type per record. When the number of errors per corrupted record increased to two, the accuracies decreased to 88.39 %, 39.54 %, 13.31 %, and 10.41 %. Figure 1 presents a full overview of the accuracy degradation for datasets corrupted with Character Edits where 10 % of all original records were corrupted.

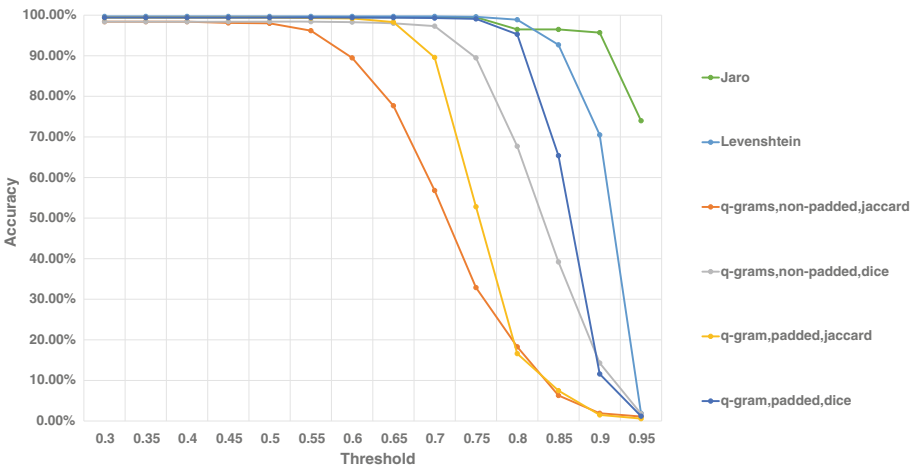

Fig. 2. Accuracy of string metrics used to perform record linkage on dataset with 10 % of the records corrupted using OCR Edits with 1 corruption per record.

4.5 Observation 5: The Efficiency Penalty for These Privacy-Preserving String Similarity Metrics is Small

The Jaro, Levenshtein, and Jaccard and Dice variants of the q-grams metric all have a $O(nm)$ time complexity, where n and m are the lengths of the strings to be compared. Because the Soundex metric is only checking for equality of the Soundex representation of the strings, its time complexity just $O(n)$. When determining whether a particular name is in the dataset, the query name is compared against all of the names in the dataset. It should be noted that the Soundex algorithm, because it is an exact rather than fuzzy match, could be made more efficient by indexing the database on the Soundex representation of the name. Also, there has been some work on eliminating the need to consider all names in the dataset when querying using the Jaro metric through the user of character and length-based filters to quickly determine if it is possible for a particular name to match the query within a specified threshold [4]. Neither of these optimizations were considered in this work.

While most of the string metrics considered have the same computational complexity, constant factors differ between the approaches. For example, because Jaro only looks for matching characters within a window that is half the length of the longer string, it is generally faster than Levenstein. To evaluate the computational efficiency of each of the string metrics in a practical setting, the time taken to perform the join operation between the original dataset and the corrupted dataset was measured. We explored the impact on the performance when the number of characters in names increases. In this case, the datasets always consisted of 10,000 records but the number of characters in each name was equal to 10, 15, or 20. These tests were done on datasets with a record corruption of 10 %, where the records were corrupted using the Character Edit technique and contained one corruption per record. The results are shown in Fig. 3. The timing

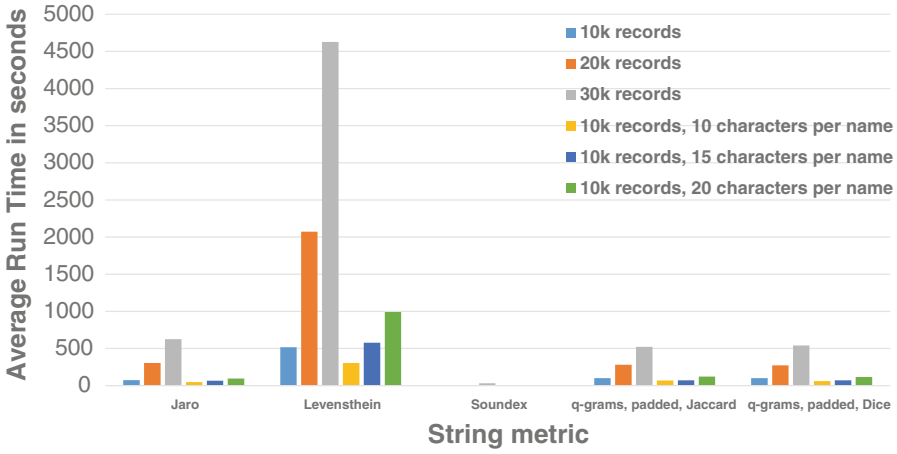

Fig. 3. The average time of record linkage using selected string metric techniques.

results of linkage performed on the other corrupted datasets were very similar to the ones presented in this figure.

The results show that the q-grams approaches are very slightly faster than Jaro in these tests, and significantly faster than Levenshtein. The average time taken to perform the join operation on the datasets using Levenshtein was more than five times the magnitude of the time taken by the other string metrics. Of course, the best speed was observed when the datasets were linked using the Soundex metric. In those cases the linkage was performed almost instantly, averaging only about one second.

Additionally, we have investigated the impact on the performance when the number of records increases. Three datasets of different volumes (10, 20, and 30 thousand records) were linked to conduct the tests. The results shown in Fig. 3 indicate that as the number of records to be linked increases, the time required to link all the records is again very similar for Jaro and the q-grams techniques, significantly greater for Levenstein, and very low for Soundex.

5 Conclusions and Future Work

In this work we evaluated the accuracy and speed of selected string metrics that support approximate matching for querying and joining databases based on encrypted names. An advanced benchmark generation tool, "GeCo", was used to produce sample datasets with records containing common mistakes in name spelling such as typographical errors, optical recognition errors, and phonetic errors. The performance of several string metrics that support approximate matching on encrypted data (four variations of q-grams based techniques and one technique based on encodings produced by the Soundex algorithm), was compared against commonly used string metrics, such as Jaro and Levenshtein, employed on unencrypted data.

Joining databases based on Soundex encodings did not prove to be a feasible option since it failed to find a correct match for more than 50 % of records when the name in a corrupted record contained one error, and for almost 75 % when corrupted records could contain two errors in a single name. Q-grams based techniques seem to be viable option for joining databases on encrypted names. While their performance in terms of precision is slightly worse than the performance of metrics such as Jaro or Levenshtein on unencrypted data, this can be easily dealt with by adjusting the threshold value that determines when two q-grams are likely to correspond to the same name.

In future work we plan to extend the range of attribute types that can be used to perform record linkage. In this study we focused on linking records based only on an individual's first and last name. However other types of attributes, such as numeric or categorical ones, can also carry PII. We want to be able to integrate those kind of attributes into private record linkage queries. Finally, we want to address the potentially significant security vulnerability of the encrypted q-grams approach, on which a frequency attack based on common q-grams can be launched, by investigating possible ways to make the encrypted q-grams resilient to those kind of attacks.

Acknowledgments. This work was partially supported by the LexisNexis corporation.

References

1. Christen, P.: A comparison of personal name matching: techniques and practical issues. In: Sixth IEEE International Conference on Data Mining Workshops, ICDM Workshops 2006, pp. 290–294. IEEE (2006)
2. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Heidelberg (2012)
3. Churches, T., Christen, P.: Some methods for blindfolded record linkage. BMC Med. Inform. Decis. Mak. **4**(1), 9 (2004)
4. Dreßler, K., Ngomo, A.C.N.: Time-efficient execution of bounded jaro-winkler distances. In: Proceedings of the 9th International Conference on Ontology Matching, vol. 1317, pp. 37–48. CEUR-WS. org (2014)
5. Giereth, M.: On partial encryption of RDF-graphs. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 308–322. Springer, Heidelberg (2005)
6. Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E., Järvelin, K.: Non-adjacent digrams improve matching of cross-lingual spelling variants. In: Nascimento, M.A., de Moura, E.S., Oliveira, A.L. (eds.) SPIRE 2003. LNCS, vol. 2857, pp. 252–265. Springer, Heidelberg (2003)
7. Muñoz, J.C., Tamura, G., Villegas, N.M., Müller, H.A.: Surprise: user-controlled granular privacy and security for personal data in smartercontext. In: Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research, pp. 131–145. IBM Corp. (2012)
8. Philips, L.: Hanging on the metaphone. Comput. Lang. **7**(12) (1990)
9. Snae, C.: A comparison and analysis of name matching algorithms. Int. J. Appl. Sci. Eng. Technol. **4**(1), 252–257 (2007)
10. Tran, K.N., Vatsalan, D., Christen, P.: Geco: an online personal data generator and corruptor. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 2473–2476. ACM (2013)
11. Vatsalan, D., Christen, P., Verykios, V.S.: An efficient two-party protocol for approximate matching in private record linkage. In: Proceedings of the Ninth Australasian Data Mining Conference, vol. 121, pp. 125–136. Australian Computer Society, Inc. (2011)
12. Yakout, M., Atallah, M.J., Elmagarmid, A.: Efficient private record linkage. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, pp. 1283–1286. IEEE (2009)