Contextual Entity Resolution Approach for Genealogical Data

Hossein Rahmani[1], Bijan Ranjbar-Sahraei[1], Gerhard Weiss[1], and Karl Tuyls[2]

[1]Maastricht University, PO Box 616, Maastricht 6200 MD, The Netherlands
{h.rahmani,b.ranjbarsahraei,gerhard.weiss}@maastrichtuniversity.nl
[2]University of Liverpool, Ashton Building, Liverpool L69 3BX, United Kingdom
k.tuyls@liverpool.ac.uk

Abstract. Due to huge amount of inaccurate information and different types of ambiguity in the available digitized genealogical data, applying Entity Resolution techniques for determining the records referring to the same entity should be considered as the first and still very important step in analysis of this type of data. Traditional methods, use a standard string similarity measure to calculate the similarity among references, neglecting the contextual information available for each reference, and then introduce the most similar pairs as matches. In this paper, first, we introduce a novel blocking strategy to reduce the number of potential candidate pairs. Second, we propose a contextual similarity measure which not only considers the string similarity among references but also contextual information available for them. Third, we evaluate our proposed method extensively from different perspectives and among many discussed patterns, the "early child death" pattern discovered to be prominent.

Keywords: Entity Resolution, Contextual Similarity, Genealogy

1 Problem Definition

The work discussed in this paper has been developed as part of a larger project, the MISS[1] (Mining Social Structures from Genealogical Data) Project, which is funded by the NWO (Netherlands Organisation for Scientific Research) Association. MISS project uses the historical certificates of BHIC[2] (Brabants Historical Information Center) to unravel the genealogical connections and also mine the social structures from a prosopographical [8] point of view.

There are three important certificate types used in this project, namely "Birth", "Death" and "Marriage". Section 2 describes these certificate types in

 In: T. Seidl, M. Hassani, C. Beecks (Eds.): Proceedings of the LWA 2014 Workshops: KDML, IR, FGWM, Aachen, Germany, 8-10 September 2014, published at http://ceur-ws.org

[1] http://swarmlab.unimaas.nl/catch/
[2] http://www.bhic.nl

detail. The digitized version of these certificates, however, is not at all flawless; many names are duplicate, have several alternative spellings, or even contain mistakes. The main challenge in this project is to find an approach which can resolve when two references refer to same entity in spite of errors and inconsistencies in the input certificates. We model this project as a system which takes large number of error-prone and inconsistent certificates as input and as an output, generates the graph of individuals in which each node represents an entity and each edge shows the family relationship among two connected entities. Two main goals of this project are 1) Detecting and eliminating duplicate references referring to same entity (Known in literature in many different ways such as Record Linkage [12, 18], the Merge/Purge prblem [6], Duplicate Detection [10, 16], Hardening Soft Databases [1], Reference Matching [9] and Entity Resolution [5, 4]), and 2) Re-constructing family relationships among individuals.

2 Input Data

We consider three certificate types "Birth", "Death" and "Marriage" as input of our system. Table 1 shows the considered features for each certificate type. As shown in Table 1 Birth certificates include 3 individual references (i.e., child, father and mother). The Death certificates include 4 individual references (i.e., deceased, father, mother and relative of deceased). Finally, the Marriage certificates include 6 references (i.e., groom, bride and father and mother of each).

Table 1: Considered features for each certificate type.

Birth Certificate	FirstName, LastName, Gender, BirthDate, BirthPlace, FatherFirstName, FatherLastName, MotherFirstName, MotherLastName
Death Certificate	FirstName, LastName, Gender, BirthDate, BirthPlace, DeathDate, DeathPlace, FatherFirstName, FatherLastName, MotherFirstName, MotherLastName RelativeFirstName, RelativeLastName, RelationType
Marriage Certificate	GroomFirstName, GroomLastName, GroomAge, BrideFirstName, BrideLastName, BrideAge, GroomFatherFirstName, GroomFatherLastName, GroomMotherFirstName, GroomMotherLastName, BrideFatherFirstName, BrideFatherLastName, BrideMotherFirstName, BrideMotherLastName

This database consists of $5,300,000$ individual references extracted from $1,170,000$ certificates (details are provided in Table 2). Considering the non-mentioned parents or relatives in some certificates, $500,000$ references do not have any name (i.e., first_name = *null* and last_name = *null*). Therefore, we have $4,800,000$ informative references. Among these references we have $170,000$

distinct first names and 100,000 distinct last names. The dates mentioned in different certificates span a period of time between 1810 and 1920. The certificates are registered in 200 different municipalities.

Table 2: Statistical information of input data.

Number of Birth Certificate	110,000
Number of Marriage Certificate	350,000
Number of Death Certificate	710,000
Number of Extracted References	5,270,000

3 Proposed Blocking Strategy

The most direct way of finding duplicates among the references is to apply pairwise comparison among the references and then consider the reference pairs with highest similarity values as a same entity. The computational order of this process is $O(n^2)$ which makes it infeasible in our project with rouphly 5,000,000 references. In order to avoid having to compare all pairs of references, we propose a new blocking strategy to split all references into different blocking partitions. This process reduces the search space and diminishes the number of potential candidate pairs.

As a blocking strategy, the previous methods use the standard string encoding systems such as Soundex [7], metaphone [14] and double-metaphone [15]. Soundex, indexes the names based on their pronunciation in English. The main goal in this algorithm is that the letter with similar pronunciations be encoded with same characters so that spelling errors can be resolved. Metaphone is an extension of the Soundex code. Compared to Soundex, this code takes into account more information about variations and inconsistencies in English spelling and pronunciation. Afterwards, Double Metaphone was proposed which takes into account spelling peculiarities of a number of other languages. This indexing algorithm generates up to two codes for each word, that can improve some of the limitations of the original Metaphone for dealing with foreign languages.

In this section, we investigate the dataset of Dutch names [13] in order to first, evaluate the effectiveness of standard string encoding systems, second, extract the informative features for blocking strategy and third, propose a blocking strategy which considers both typing error and conventional name variations in Dutch names. The following subsections discuss in details the three mentioned steps.

3.1 Dutch Name Dataset

There are different writing variations for each (Dutch) name. For example, “Ghendrik”, “Haendrik”, “Handrikus”, “Hanri” and “Hedrik” are all referring to the same entity “Hendrik”. The reason behind this name variations could be of typing error or some historical/geoghraphical issues. In this section, we use the dataset of Meertens Institute [13] to find the relationships among different

variations of Dutch names with their standard format. In total, the Meertens database contains 44,000 distinct first names (18,000 and 26,000 for male and females, respectively) and 120,000 distinct last names. The main attributes of the dataset are *name*, *standard name* and *popularity*. In this dataset, in average each standard first name has about 16 name alternatives while the standard last names have about 8 alternatives in average. However, there exists some standard names with very high number of alternatives.

3.2 Extracting Informative Features

So far, we have analyzed the following features from the the Meertens dataset.

F1: [Boolean feature] If first 2 letters of name and standard name are equal.
F2: [Boolean feature] If first 3 letters of name and standard name are equal.
F3: [Boolean feature] If last 2 letters of name and standard name are equal.
F4: [Boolean feature] If last 3 letters of name and standard name are equal.
F5: [Boolean feature] If size of name and standard name are equal.
F6: [Integer feature] Absolute difference of name length and standard length.
F7: [Integer feature] Number of longest first equal chars.
F8: [Integer feature] Number of longest last equal chars.
F9: [Boolean feature] If *soundex* code of name and standard name is equal.
F10: [Boolean feature] If *metaphone* code of name and standard name is equal.
F11: [Boolean feature] If *double-metaphone* code of name and standard name is equal.
F12: [Integer feature] Longest common chars between name and its standard name.

Table 3 calculates the *min*, *mean*, *s.t.d.* and *max* for each feature for the *male first name*, *female first name* and *last name* datasets.

Table 3 provides detailed information about the 12 very basic and important features of Dutch names. Among all the features, **F1** is a very discriminative feature as it is true in more than 70% of the cases (i.e., the first two letters of a name and its standard name are equal in more than 70% of the cases). Among the phonetic-based string similarity measures (**F9**, **F10** and **F11**) Soundex code has the highest score of being identical between name and its standard form in about 50% of the cases. However, the absolute difference of name length and its standard form length **F6** has a maximum of 15, which means some name lengths can deviate very much from length of its standard form.

In next subsection, we use the most discriminative features discussed in this section to build a blocking key for partitioning the references based on their similarities.

3.3 Blocking Key Generation

Following the conclusions discussed in Section 3.2, we propose a new blocking key strategy which considers both name variations and spelling error.

Table 3: Feature analysis of Dutch names. Features **F6**, **F7**, **F8** and **F12** are continuous features and the rest of features are all binary.

	first name (male)				first name (female)				last name			
Feature	min	mean	s.t.d.	max	min	mean	s.t.d.	max	min	mean	s.t.d.	max
F1	0	0.71	0.46	1	0	0.70	0.46	1	0	0.79	0.40	1
F2	0	0.52	0.49	1	0	0.50	0.49	1	0	0.60	0.48	1
F3	0	0.36	0.49	1	0	0.42	0.49	1	0	0.54	0.50	1
F4	0	0.27	0.44	1	0	0.30	0.45	1	0	0.45	0.49	1
F5	0	0.35	0.48	1	0	0.34	0.48	1	0	0.43	0.5	1
F6	0	1.15	1.31	15	0	1.10	1.31	13	0	0.77	0.88	10
F7	0	2.90	2.07	13	0	2.90	2.06	11	0	3.57	2.41	16
F8	0	1.57	2.07	13	0	1.77	2.06	11	0	2.59	2.65	16
F9	0	0.50	0.5	1	0	0.47	0.5	1	0	0.58	0.49	1
F10	0	0.31	0.47	1	0	0.29	0.46	1	0	0.42	0.49	1
F11	0	0.39	0.49	1	0	0.37	0.49	1	0	0.49	0.49	1
F12	0	3.90	1.85	14	0	3.98	1.85	12	0	4.82	2.1	17

$$\begin{aligned}\textsc{Blocking\_Key}(r_i) = \textsc{Gender}(r_i) \\ +\textsc{FirstName}(r_i)[:3] + \textsc{FirstName}(r_i)[-2:] \\ +\textsc{LastName}(r_i)[:3] + \textsc{LastName}(r_i)[-2:] \\ +\text{soundex}(\textsc{FirstName}(r_i)) + \text{soundex}(\textsc{LastName}(r_i))\end{aligned} \quad (1)$$

where in Formula 1, STRING[:i] and STRING[-i:] refers to the first i and last i characters of the STRING, respectively.

For each reference in our dataset, we build its blocking key and then we assume all the references with similar blocking key in the same block. This process builds blocks with different sizes (=member count). As the size of one block increases our confidence for that block decreases. Formula 2 calculates the Confidence value for block b_i.

$$\text{Conf}(b_i) = \frac{N}{\text{size}(b_i)} - 1 \quad (2)$$

In this formula, N is the number of all references and size(b_i) returns the number of references belong to block b_i. In the most extreme case, all references belong to one block and the confidence of that block becomes 0 (i.e., Conf(b_0) = $N/N - 1 = 0$).

In the next section, we propose a contextual similarity measure to compare all reference pairs belonging to similar blocks.

4 Contextual Similarity Measure

After partitioning all the references into different blocks, now we could compare all the reference pairs belonging to the same block. The existing similarity

measures such as Levenstinen, Jaro-Winkler, etc. [17, 11, 2] simply calculate the string similarity between reference's First and Last Names neglecting the contextual information available for each reference.

As discussed in Section 2, the references that are used in this paper are extracted from three different type of certificates: "Birth", "Death" and "Marriage". Each of these certificate types contains the information of a group of Subfigs. 2(a) and 2(b) show that which should be considered when we compare two references. For example, imagine two arbitrary references r_i and r_j where both belong to the same block b_k (i.e., $r_i \in b_k$ and $r_j \in b_k$). If their partners both belong to another block b_L (i.e., partner$(r_i) \in b_L$ and partner$(r_j) \in b_L$) then the probability that r_i and r_j referring to same entity should be increased, comparing to the case that their partners belong to different blocks. To consider the contextual information hidden in each certificate, we use Formula 3 to extract the Block Context of each certificate c_i.

$$BC(c_i) = \bigcup_{r_j \in c_i, r_j \in b_k} b_k \tag{3}$$

$BC(c_i)$ simply includes the block ids of all reference members of certificate c_i. We propose a following similarity measure to consider both String similarity among references in addition to their certificate contextual information.

$$Similarity(r_i, r_j) = Sim_{NC}(r_i, r_j) + Sim_{BC}(r_i, r_j) \tag{4}$$

In Formula 4, $Sim_{NC}(r_i, r_j)$ and $Sim_{BC}(r_i, r_j)$ calculates the "No Context" and "Blocking Context" similarity values between two references r_i and r_j, respectively. In, Formula 5, we use Jaro-Winkler algorithm [17] to calculate the string similarity between FirstName and LastName of two references r_i and r_j.

$$\begin{aligned} Sim_{NC}(r_i, r_j) = \frac{1}{2}\Big[JaroWinkler(\textsc{FirstName}(r_i), \textsc{FirstName}(r_j))\Big] \\ + \frac{1}{2}\Big[JaroWinkler(\textsc{LastName}(r_i), \textsc{LastName}(r_j))\Big] \end{aligned} \tag{5}$$

If two references r_i and r_j belong to two certificates c_i and c_j respectively, then we use Formula 6 to calculate the contextual-based similarity between them.

$$Sim_{BC}(r_i, r_j) = \frac{\sum_{b_k \in \{BC(c_i) \cap BC(c_j)\}} \text{Conf}(b_k)}{\sum_{b_k \in \{BC(c_i) \cup BC(c_j)\}} \text{Conf}(b_k)} \tag{6}$$

Formula 6 checks if the other references in the same certificates belonging to the similar blocks or not.

5 Empirical Studies

In this section, first, our proposed blocking strategy is applied on the genealogical dataset introduced in Section 2. Then, the contextual similarity measure

proposed in Section 4 is used to extract the links between certificates. Finally, by means of examples and manual evaluation, the final results are evaluated by domain experts.

5.1 Results of Proposed Blocking Technique

We applied our proposed blocking strategy to the BHIC dataset, which contains about $5,000,000$ references. As a result, $690,000$ blocks are constructed with different sizes ranging from size 1 to $3,845$, where each of the blocks of size one contains just 1 reference and the block with largest size contains $3,845$ references; the block key of this largest block is "female_Mar_Jan _ia_en_M600_J525" which turns out to be the most pattern among Dutch references reported between 1890 and 1920. The average block size is 7 and the standard deviation is 29. This shows that not many blocks of very large size exist. Fig. 1 shows the block size distribution by focusing on the blocks with size 2 to 50.

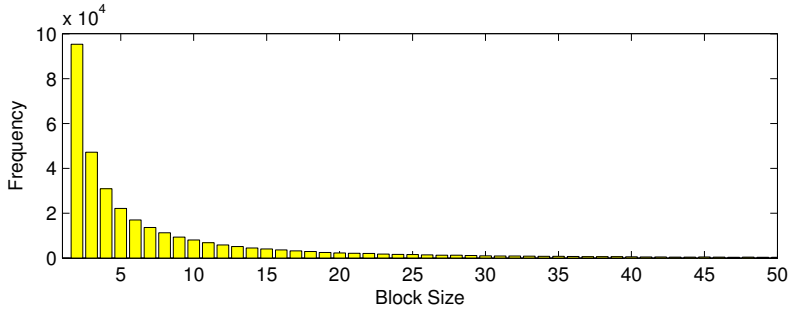


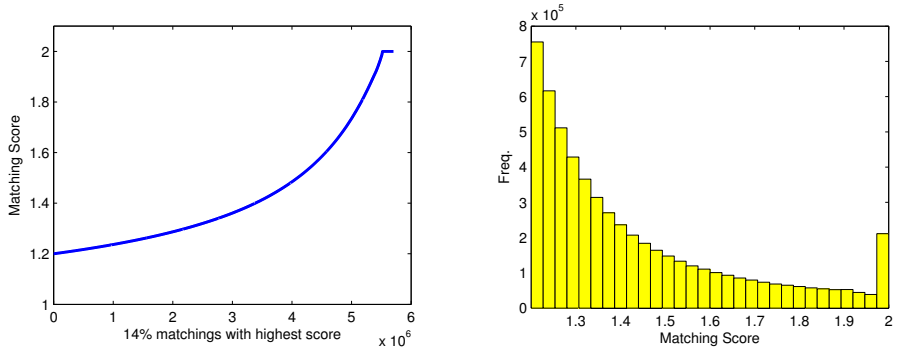
Fig. 1: Distribution of block sizes. The average block size is 7 and the standard deviation is 29. Blocks with size 1 are excluded from the figure as they contain just 1 reference and are not informative in matching.

Given that originally about 25×10^{12} pairwise comparisons (i.e., $5,000,000 \times 5,000,000$) were required for accomplishing the task of traditional entity resolution, based on the partitions introduced by the proposed blocking strategy, the average search space is now reduced by 3.5×10^{-5} (i.e., $\frac{690,000 \times 7^2}{25 \times 10^{12}}$).

5.2 Result of Contextual Similarity Measures

In this subsection, we report the results of applying the proposed contextual similarity measure on matching candidates from all blocks b_i of size less than or equal to 100 (i.e., $size(b_i) =< 100$). This generates in total $40,489,999$ matching pairs with similarity scores less than 2.0, where 14% of matching candidates (i.e., $5,703,687$ pairs of references) have a score larger than 1.2. The score distribution of these matching candidates are shown in Fig. 2.

Subfigs. 2(a) and 2(b) show that many of the matching candidates have a matching score equal to 2.0. We consider these matches as perfect matches (identical certificates) which can refer to either record duplicates or correct matches where the same group of references are mentioned in both certificates.



(a) (b)

Fig. 2: The contextual similarity measure of 14% matching candidates with highest score (a) the matching candidates are sorted based on the matching score. 185,000 of the candidates have a perfect match with exact matching score 2.0. (b) distribution of the matching candidates with a score higher than 1.2.

Table 4 provides a categorization of all the certificate-pairs that contain the 185,000 perfect matches with score = 2.0 (i.e., about 90,000 certificate pairs). Table 4 shows that more than 50,000 matches refer to connections between death certificates. By further exploration of the results we realized that about 28% of these matches refer to record duplicates[3]. Besides 78% of the matches refer to the certificates with an average of 2.1 year difference in issue time. One major reason for these matches can be early death of child in families of 19^{th} century which results in using the same name for next child which might end with death of next child as well.

By exploring the 28,000 matches between death and birth certificates, again 25% of these matches can be because of the early death of the birth as both certificates are issued in the same place in the same year. However, 75% of the matches refer to the matches of birth and death of an entity with an average age of 28.4 years. The reason for having such a low average age is that in such death certificates, no relative name is mentioned for the deceased person (i.e., most probably the deceased has been single), which is the case for young references. For details about other matching types refer to Table 4.

5.3 Result of Contextual Matching

In this subsection, we discuss the results of the proposed contextual matching technique by presenting some examples of the revealed matches between different references, and also the results of a manual check of over 300 instances of data will be provided.

[3] record duplicates in our genealogical dataset refer to the cases that an event is issued by two authorities or due to data storage inconsistencies the record is stored more than one time with minor differences in location name, archive index, etc.

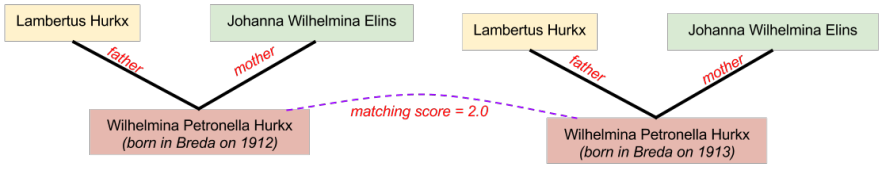
Table 4: Categorisation of matches with score 2.0 where the number of certificates seen for each type of matching and the average difference in date of certificate issue is provided.

Matching Type	Avg. Date Diff.	Freq.
Death Certificate *(to)* Death Certificate • *28% due to record duplicate* • *72% due to early child death and using the name for next child, or other reasons.*	2.1 years	51,293
Death Certificate *(to)* Birth Certificate • *25% due the early child birth* • *75% due to matching between birth and death of an entity, or others*	9.8 years	28,401
Birth Certificate *(to)* Birth Certificate • *100% due to early child death and using the name for next child, or others*	3.5 years	8,679
Marriage Certificate *(to)* Marriage Certificate • *100% due the record duplicate*	0 years	1642
Marriage Certificate *(to)* Death Certificate • *100% due to matching between a death and marriage certificate of an entity when the parents of deceased partner are not mentioned in the marriage certificate*	26 years	99

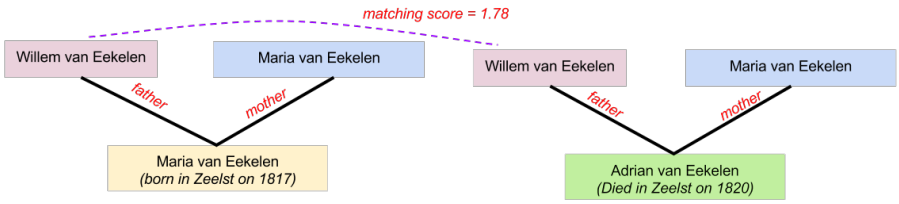
By a careful study on discovered matchings with highest score, different patterns can be introduced, where for each pattern specific certificate roles are matched with each other. For instance, one common pattern can be a match between two born references due to similarities between their names and their parents' names. Subfig. 3(a) illustrates an example where the match between two born references is considered as a perfect match with score = 2.0. In this example, both certificates are issued in the same place with one year difference in time of issue. Therefore, this case might refer to an early death of a born chid, where her name is used for the the next born child in the following year.

Another common pattern in discovered matches is the high contextual similarity between parents of a birth or death certificate and parents of another birth or death certificate. Subfig. 3(b) shows an example of this pattern where a father in a birth certificate is matched with father in a death certificate which is issued three years later in the same place. The matching is not perfect (score = 1.78) as the children have different first names (this can refer to two siblings).

In order to evaluate the quality of the revealed matches between references, we chose 324 matches randomly (from matches with score higher than 1.3) and for each match a domain expert, familiar with genealogical data, used the provided evidence (names of references, family relations, place and date of issue, blocking key and blocks confidence) to evaluate the match by choosing either a



(a) Matching between two birth certificates (score = 2.0)



(b) Matching between a birth certificate and a death certificate (score = 1.78)

Fig. 3: Examples of matched pairs with different scores (a) A perfect match between two birth certificates, which are issued in the same place with one year difference.This can be due to early death of the first child, and using her name for next born child. (b) Father in a birth certificate is matched to the father in death certificate of another child 3 years later. (As can be seen the born child in left certificate has an identical name with mother)

True Positive or a *False Positive* category[4]. This evaluation approach is similar to the approach described in [3]. Fig. 4 depicts the distribution of true positive and false positive matches for manual evaluation. In this figure, no evidence of false positive matches with a score higher than 1.7 can be seen while for lower scores many false positive matches are seen.

6 Conclusions and Future Work

The reliability of any data analysis method strongly depends on the quality of the input data. Considering the Genealogical data, with huge amount of inaccurate information and different types of ambiguities, applying Entity Resolution techniques for cleaning and integrating the references extracted from different historical certificates should be taken into account as the first step toward any data analysis approach. Traditional methods, use a standard string similarity measure to calculate the similarity among references, neglecting the contextual

[4] Please note that due to missing data, typing errors, redundancies and lack of extra evidence confirming that two references point the same real entity is impossible in many of the cases. Therefore, in evaluations of this paper, we stick to the evidence at hand and assume that a reasonable similarity between two references, similar family members, and feasible date and similar places can suggest a true positive match, otherwise it will be considered as a false positive match.

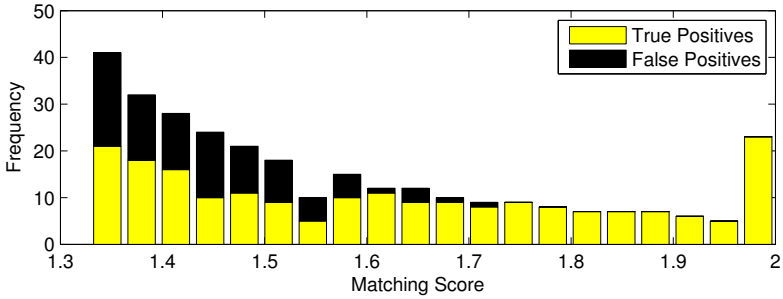


Fig. 4: Distribution of True/False positive matches using a manual evaluation over 300 matching candidates. The results show 70% True positives and 30% False positives. The false positives are detected for scores lower than or equal to 1.7.

information available for each reference, and then introduce the most similar pairs as matches. In order to avoid having to compare all pairs of references, we investigated the dataset of dutch name [13], we selected the most discriminative features and finally, we proposed a blocking strategy to split all references into different blocking partitions. As a result of applying our proposed blocking strategy, the search space is reduced by 3.5×10^{-5}. To compare all the references belong to the similar blocks, we proposed a contextual similarity measure which not only considers the string similarity among references but also contextual information available for them. According to considered genealogical certificates, we defined context of each reference as its first level family relationships (i.e., partner, father, mother etc) and accordingly, we increased the probability of reference matches if they share a common context. We evaluated our proposed contextual similarity measure from different perspectives and among many discussed patterns, the "early child death" pattern discovered to be prominent. In this pattern, child dies in early years and the family uses the same name for the next born baby.

Regarding future research induced by our work, we see three particularly important directions for refinement and extension of our approach. First, further exploration of possibilities for extensive validation of the achieved results. This is challenging because we typically do not have grounded truth against which the results can be directly compared. We have already discussed and validated our results in Sections 5.2 and 5.3 by human domain experts; however, it would be very useful and considerably more efficient to have a way of (at least partially) evaluating the results automatically or at least semi-automatically by simulating domain-expert behavior. Second, investigation of Random Walk to take into account a wider range of contextual information such as second-level family members. And third, when the graph of entities is built, the study of common characteristics of specific groups of entities in order to unravel previously unknown information and connections within the groups [8].

Acknowledgments. This research has been supported under the NWO CATCH program in the MISS project (project no. 640.005.003). The authors are grateful to the BHIC center for the support in data gathering and direction.

References

1. William W. Cohen, Henry A. Kautz, and David A. McAllester. Hardening soft information sources. In Raghu Ramakrishnan, Salvatore J. Stolfo, Roberto J. Bayardo, and Ismail Parsa, editors, *KDD*, pages 255–259. ACM, 2000.
2. Julia Efremova, Bijan Ranjbar-Sahraei, and Toon Calders. A hybrid disambiguation measure for inaccurate cultural heritage data. In *the 8th Workshop on LaTeCH*, pages 47–55, 2014.
3. Julia Efremova, Bijan Ranjbar-Sahraei, Frans A Oliehoek, Toon Calders, and Karl Tuyls. An interactive, web-based tool for genealogical entity resolution. In *25th Benelux Conference on Artificial Intelligence*, pages 376–377, 2013.
4. Julia Efremova, Bijan Ranjbar-Sahraei, Frans A Oliehoek, Toon Calders, and Karl Tuyls. A baseline method for genealogical entity resolution. In *Workshop on Population Reconstruction*, 2014.
5. Lise Getoor and Ashwin Machanavajjhala. Entity resolution for big data. In *Proceedings of the 19th ACM SIGKDD*, pages 1527–1527. ACM, 2013.
6. Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. *SIGMOD Rec.*, 24(2):127–138, May 1995.
7. Donald E. Knuth. The art of computer programming 1: Fundamental algorithms 2: Seminumerical algorithms 3: Sorting and searching, 1968.
8. Verboven Koenraad, Carlier Myriam, and Dumolyn Jan. A short manual to the art of prosopography. In Keats-Rohan K.S.B., editor, *Prosopography Approaches and Applications. A Handbook*, pages 35–69. Unit for Prosopographical Research (Linacre College), Oxford, 2007.
9. Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD*, pages 169–178. ACM, 2000.
10. Alvaro E. Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *DMKD*, pages 0–, 1997.
11. Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.
12. Howard B. Newcombe, James M. Kennedy, S.J. Axford, and A.P. James. Automatic Linkage of Vital Records. *Science*, 130(3381):954–959, October 1959.
13. Meertens Institute Databases of Names. `http://www.meertens.knaw.nl/cms/en/collections/databases`. Accessed 2014-06-27.
14. Lawrence Philips. Hanging on the metaphone. *Computer Language*, 7(12), 1990.
15. Lawrence Philips. The double metaphone search algorithm. *C/C++ users journal*, 18(6):38–43, 2000.
16. Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD*, KDD '02, pages 269–278, New York, NY, USA, 2002. ACM.
17. William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.
18. William E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*, 1999.