Entity Matching: A Case Study in the Medical Domain

Luiz F. M. Carvalho[1] and Alberto H. F. Laender[1] and Wagner Meira Jr.[1]

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
{fernandocarvalho,laender,meira}@dcc.ufmg.br

Abstract. In this paper, we propose a simple and effective solution for the *entity matching* problem involving data records of healthcare professionals. Our method depends on three attributes that are available in most data sources in the medical domain: *name*, *specialty* and *address*. We apply a blocking technique to avoid comparisons, three matchers for conciliating the data records and a rule-based heuristic to combine the matchers. We performed experiments involving data from three Brazilian Web sources of healthcare professionals. Our results show that our solution is able to avoid unnecessary comparisons and provides good results.

Keywords: Entity Matching, Medical Records, Blocking, Matchers.

1 Introduction

With the increasing amount of data available on the Web, especially in social media networks and online catalogs of products and services, a recent challenge in Computer Science is the identification of data records related to the same real world entities. For example, a product offered on a website may appear in another one under a different name or description. Furthermore, multiple websites with similar functionality may offer different products. The conciliation of such products is a core task towards price monitoring [1, 14].

The task of disambiguation is also crucial in scenarios in which it is necessary to map all records related to a person considering her personal information such as name, occupation and address. In these cases, the task can be very complex due to existence of homonyms, incomplete data and multiple ways to represent the same information. Some examples of scenarios in which this task is performed are gathering all profiles of a same person in multiple social networks [19] and collecting all papers or co-authors of a same author [9, 21].

The task of integrating all records related to a given person is known as *entity matching* (also called *data matching*, *entity resolution* and *record linkage*) and can be defined as [15]:

Given two datasets A and B from two semantically related data sources S_A and S_B, the entity matching problem consists in finding all matches between data

records in $A \times B$ that refer to the same entity in the real world.

A similar problem is *duplicate detection* in which the matching is performed over records from the same dataset [4, 5, 13].

Due to today's widespread use of medical management systems and the popularization of medical service websites, the medical domain is rich in applications that demand entity matching solutions for the integration of records of healthcare professionals. This task is important for many reasons that range from providing better search facilities to fraud detection.

In this paper, we propose a problem-oriented method for integrating data from healthcare professionals available on distinct Web sources, which is based on simple string matching strategies and adopts an overlapping blocking mechanism for reducing the number of comparisons between the records. For its assessment, we use data from three Brazilian Web sources: two general purpose healthcare professional directories, *Apontador*[1] and *Doctorália*[2], and the National Directory of Health Establishments maintained by the Brazilian Ministry of Health, *CNES*[3]. In summary, the main contributions of this paper are:

- A simple and effective solution for integrating data from healthcare professionals;
- A case study involving real data composed of 406,564 records collected from three distinct data sources;
- A detailed discussion over the main implementation issues and decisions involved in our method for entity matching in the medical domain.

Our method is generic for the medical domain and depends only on attributes that are usually available in most Web sources that provide information on healthcare professionals.

The rest of this paper is organized as follows. Section 2 provides some background on the core tasks involved in the entity matching problem. Section 3 presents our method and describes its implementation details. Section 4 discusses our experimental results. Finally, Section 5 presents our conclusions and some insights for future work.

2 Background

According to Christen [5], the process for solving the *entity matching* problem can be divided into five steps: (i) data pre-processing, (ii) indexing, (iii) record-pair comparison, (iv) classification and (v) evaluation. The first step involves the data preparation to ensure a standardized formatting for all involved data sources. The following three steps comprise the main tasks involved in the actual process of comparing the data records, whereas the last one consists of evaluating the quality of the results. Next, we briefly provide some background on

[1] *http://www.apontador.com.br*
[2] *http://www.doctoralia.com.br*
[3] *http://cnes.datasus.gov.br*

specific tasks, namely *blocking*, *matcher selection* and *matcher combination*, that support the three main steps of the entity matching process. For more details, we refer the reader to [5] and to some surveys on entity matching found in the literature [3, 8, 15].

Blocking. The purpose of blocking is to reduce the number of comparisons among the data records. It consists of defining a strategy to group the records in blocks so that only pairs of records in the same block are compared, avoiding the quadratic cost of comparing all pairs. A good blocking strategy minimizes the number of comparisons but does not separate related records into different blocks. A blocking strategy can either assign each record to just one block or set multiple blocks for each record (blocking with overlapping). In [11], the authors propose the *Sorted Neighborhood Method*, which is considered one of the first blocking methods described in the literature. The *Canopy Clustering* method [17] provides a clustering-based approach for blocking. For more details, Draisbach and Naumann [7] present a comparison of state-of-the-art blocking methods.

Matcher Selection. Matchers are algorithms that measure the similarity between a pair of records. There are two kinds of matchers: *context-based* and *value-based* matchers. Context-based matchers use structural information, such as graph distances, to define the similarity of a pair of records. A popular application of context-based matchers is the disambiguation of authors through co-authorship graph analysis, such as the solution proposed in [12]. On the other hand, value-based matchers establish a similarity degree between a pair of records by using only attribute values. The most popular value-based matchers are based on string comparison. For example, the *Fragment Comparison* algorithm [18] compares the components of author names as they appear in bibliographic citation records in order to decide whether they refer to the same author or not. This algorithm is robust to typing errors, abbreviations and variations in the sequence of name components. A comparison of string metrics for matching names and records is presented in [6].

Matcher Combination. This task consists in combining the values resulting from several distinct matchers to decide whether or not a pair of records is related to each other. The most popular solutions for combining matchers can be classified into three types: numerical, rule-based and flow-based. Numerical combinations apply a mathematical function to combine the involved matchers, as described in [10] and [20]. Rule-based combinations rely on logical operations and thresholds specifically defined for each case [2, 16]. Finally, flow-based combinations involve complex rules and many steps to perform the combination, like in the *MOMA* method [22].

We have implemented simple and effective solutions for each one of the above tasks, as described in the next section.

3 Proposed Method

In this section, we describe our solution for the entity matching problem in the medical domain. Our method relies on three specific attributes, *name*, *specialty* and *address*, of which *name* plays a major role in the matching process. We start by introducing the concept of *name relevance* that is central to our method. Then, we describe our solution for the three main tasks involved in the matching process discussed in the previous section: blocking, matcher selection and matcher combination. Finally, we discuss some strategies to reduce the execution cost of the whole process.

Name Relevance. To define the concept of name relevance, we consider that a person's name consists of several components. For example, the name "*luiz fernando magalhaes carvalho*" consists of four components: "*luiz*", "*fernando*", "*magalhaes*" and "*carvalho*". Informally, we can say that the relevance of a person's name depends on the discrimination power of each one of its components. Thus, let D be the set of datasets being matched and N the set of all names found in any dataset in D. For each name component c in N, its relevance is a value between 0 and 1 given by the ratio of its frequency and the frequency of the most frequent name component r in N, as expressed by:

$$relevance(c) = frequency(c)/frequency(r)$$

The relevance of a name C is given by the sum of the relevance of each of its components c_i. If the resulting value is greater than 1, it is set to 1, so that the relevance of each name is also a value between 0 and 1, as expressed by:

$$relevance(C) = \sum_{c_i \ \in \ C} relevance(c_i)$$

$$relevance(C) = \begin{cases} relevance(C), \ if \ relevance(C) < 1 \\ 1, \ otherwise \end{cases}$$

As we show next, the relevance of a name component is used to avoid unnecessary comparisons within a block whereas the relevance of a (full) name is used in the matcher combination.

Blocking. Blocking is a strategy to group records in blocks so that only records in the same block are compared. The blocking strategy of our method consists in creating one block for each name component found in any data source in D. As each block B is associated with a name component c, B contains all records that include a name with a name component c. Thus, this is a blocking strategy that implements an overlapping scheme, as each record is assigned to all blocks related to its name components, which means that each record is compared with all records that have at least one name component in common. This strategy is based on the assumption that if two records are related, they have at least one

name component in common.

Matcher Selection. Having blocked all records, we need to use a matcher to measure the similarity of each record pair. For each one of the three attributes considered (*name*, *specialty* and *address*), we have implemented a matcher that returns a score between 0 and 1. Thus, for each record pair, three specific scores are produced: *name similarity score*, *specialty similarity score* and *address similarity score*. These scores are then combined in order to define whether the records are related or not. Since all three attributes are strings, their respective matchers are based on string comparison strategies empirically chosen for this specific purpose. Considering that the matching strategy for the attributes *specialty* and *address* are simpler, we describe them first.

Our matchers for the attributes *address* and *specialty* are based on the *Levenshtein* edit distance. Their returning scores have been defined as the complement of this metric, which measures the rate of changes that must be performed in two strings to make them identical. For example, given a pair of addresses $A1$ and $A2$, the corresponding *address similarity score* is computed as:

$$Levenshtein\_distance(A1, A2) = 1 - Levenshtein\_distance(A1, A2)$$

Unlike the address and specialty matchers, the name matcher is more complex and comprises four steps. Since names usually present typing errors, abbreviations, minor variations and missing components, we need a more sophisticated strategy to be able to properly match them. In addition, the first and last names are usually more reliable than the other names component and should be accordingly weighted. Thus, for our name matcher, we have developed a four step heuristic algorithm based on the *Fragment Comparison* algorithm [18] that satisfies all these conditions. The first three steps of the algorithm compare, respectively, the first name, the last name and the other names components of each pair of records, producing three similarity scores that range from 0 and 1. Then, the last step combines the three scores.

The first name and last name scores are similarly computed based on the *Levenshtein* edit distance. If the first (last) name in both records is not abbreviated, the first (last) name score is the *Levenshtein* similarity between them. On the other hand, if the first (last) name is abbreviated in one or both records, the first (last) name score is 1 if the first letter in both name components is the same or 0 otherwise. The similarity score for the other names components is computed as the rate of matched names among the remaining names. A match occurs when either two of the remaining names have a *Levenshtein* similarity above 0.8 in the case of non-abbreviated names or the first letters are the same in case of abbreviations. As the priority is the matching of non-abbreviated names, it is performed first, as shown in Algorithm 1.

Finally, after we have the individual scores for first name, last name and other names, the full name similarity score is computed as a linear combination of these three scores. We consider that the other names score is less relevant than the first and last name scores and set its weight to a lower value. We

Algorithm 1 Other names comparison step in the name comparison matcher.

```
numberRemaining ← size(name1) + size(name2) − 4
for all x ∈ otherNames(name1) do
  for all y ∈ otherNames(name2) do
    threshold ← 0.8
    if (matched(x) == 0) and (matched(y) == 0) then
      if (length(x) > 1) and (length(y) > 1) then
        if Levenshtein(x, y) ≥ threshold then
          matched(x) ← 1
          matched(y) ← 1
          matches ← matches + 2
for all x ∈ otherNames(name1) do
  for all y ∈ otherNames(name2) do
    if (matched(x) == 0) and (matched(y) == 0) then
      if (length(x) == 1) or (length(y) == 1) then
        if firstLetter(x) == firstLetter(y) then
          matched(x) ← 1
          matched(y) ← 1
          matches ← matches + 2
other names score ← matches / numberRemaining
```

then use entropy to calibrate the weights of the first and last name scores, based on the intuition that a name component with larger uncertainty is more relevant.

Matcher Combination. After we have compared each pair of records in each block, we decide whether they are related or not based on the three similarity scores computed by the matchers and on the relevance of the corresponding full names. The solution proposed is a heuristic based on logical operations and thresholds that have been empirically defined. In the next section we show the details of the combination algorithm.

Pruning Comparisons. We also propose two heuristics for pruning unnecessary comparisons. The first heuristic consists in discarding blocks that are associated with low relevance name components. A threshold B defines the number of blocks to be discarded, i.e., the number of blocks for which no comparison is performed between their records. As stated before, each block is associated with a name component and for each such name we have computed its relevance. The B skipped blocks are those associated with names with small relevance. If a name is too popular, its respective block is large and the cost of comparing all its record pairs is huge. In addition, a popular name presents very low discrimination power, so it is likely that the matching rate within its block is very low. The second heuristic does not compare two records if the similarity score between two names is below a threshold S, that is, it does not compare the other two attributes, *specialty* and *address*. If the name similarity score produced by each matcher is smaller than S, it is very unlikely that the records are related, so it is not necessary to proceed with the remaining comparisons and the matcher combination. In the next section we describe how the values of B and S have been chosen.

4 Experimental Evaluation

4.1 Experimental Setup

Data Collected. We collected data about Brazilian healthcare professionals from three data sources: *Apontador*, *Doctorália* and *CNES*. After collecting the data, we extracted the attributes *name*, *specialty* and *address*. We selected only records of professionals from the state of Minas Gerais in order to reduce the data volume and make the calibration process easier. We believe that there is no significant difference between the data distribution among Brazilian states and our results should be closer to those from the whole country. Table 1 shows the total number of records collected from each data source.

Table 1. *Total number of records collected from each data source.*

Data Source	*Apontador*	*Doctorália*	*CNES*
Records Collected	14,060	10,324	382,180

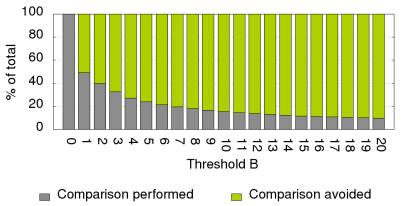
Blocking Parameters. The ideal value of B should minimize the number of comparisons but avoiding to separate related records that have in common only names associated with the largest B blocks. Therefore, if the value of B is too small, it does not avoid unnecessary comparisons. On the other hand, if the value of B is too large, some similar pairs may not be compared, although several comparisons are avoided. We then performed the following experiments.

First, we measured the amount of comparisons avoided by each value of B. Fig. 1(a) shows the percentage of comparisons avoided as a function of B considering that if B is equal to zero, no comparison is avoided, resulting in more than 11.6 billion comparisons. The results show that the rate of comparisons decreases fast initially, but the pace slows down in the range between $B = 5$ and $B = 14$. As it is not clear the stabilizing value, we chose the value $B = 5$ that avoids 76% of the original number of comparisons.

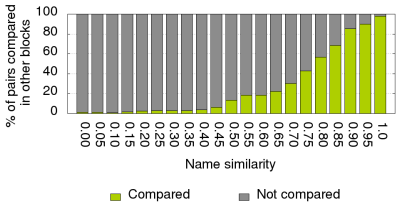
Next, we verified whether $B = 5$ was appropriate. Fig. 1(b) shows, for each similarity name range, the percentage of record pairs pruned for $B = 5$ that were also in another block and, therefore, are compared anyway. The results show that the larger the name similarity, the larger the number of comparisons. Thus, we can conclude that the threshold $B = 5$ prunes unnecessary comparisons and guarantees that pairs with related names are compared in non-pruned blocks.

According to Fig. 2(a), which shows the similarity distribution for the attribute *name*, the frequency of similar names decreases fast from 0.6, indicating that this is a good threshold to separate the actual similar names from random matches. Thus, S was set to 0.6.

Matchers and Combination Algorithm. As already described, for each pair of records compared, the matcher combination algorithm returns a binary value

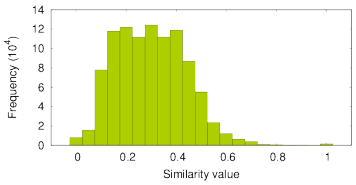


(a) Rate of comparison avoided

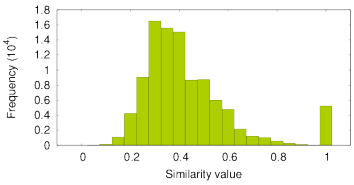


(b) Rate of skipped pairs compared in other blocks

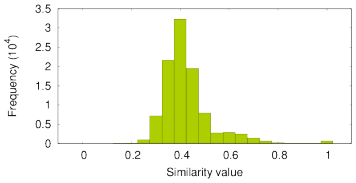
Fig. 1. Experiments results for choosing the value of B.



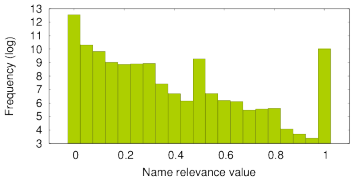
(a) Name similarity distribution



(b) Specialty similarity distribution



(c) Adress similarity distribution



(d) Full name relevance distribution

Fig. 2. Distribution of the scores considered in the algorithm for matchers combination.

indicating whether the two records match. Its implementation considers the similarity scores of the attributes *name*, *specialty* and *address*, and the relevance of the respective full names.

We start by defining the parameters for determining the matching score of a name. As already mentioned, we consider the first and last name components more relevant and thus set the weight of the other names component to 0.3. We then measure the entropy of the first and last name components, and set their weights based on the assumption that the higher their entropy, the higher their weight should be. We have found an entropy of 1.15 for the first name component and of 0.7 for the last name component. Thus, we proportionally set their weights to 0.43 and 0.27 respectively, resulting in the following expression for the name similarity score:

$$nameScore = ((0.43 \times scoreFirstName) + (0.27 \times scoreLastName) + (0.3 \times scoreOtherNames))$$

Fig. 2(a)-(c) show the distribution of the similarity scores of the three attributes for a random sample of one million pairs. Note that for the attributes *specialty* and *address*, the sample also present a name similarity above 0.6. We also observe that the highest frequencies are associated with values smaller than the similarity threshold, showing the effectiveness of our heuristic. We have also measured the correlation between the matchers' returned values, but the results were smaller than 0.1, indicating that there was no correlation among them. As a consequence, the thresholds for each attribute type may be independently defined and set to a different value. We then present a histogram of the relevance scores for the 406,564 full names (Fig. 2(d)) extracted from our datasets. We can see that the threshold for name relevance should be lower to avoid missing relevant pairs, but we still have a large number of irrelevant pairs associated with lower scores.

Our heuristic strategy was empirically set by sampling the results. It is based on logical operators and thresholds, and considered the following assumptions:

- We divided all full names in the dataset into two groups: relevant names and non-relevant names. We consider a name as relevant if its relevance value is equal or greater than 0.2. This threshold sets 15% of the names as relevant.
- We set a threshold for each one of the three matchers in order to separate record pairs with similar attribute values.
- We set specific thresholds for the attribute *name* considering the cases in which the names are relevant and not relevant. For relevant names, the threshold is 0.8. Otherwise, it is set to 0.9.
- For the *specialty* and *address* attributes, the thresholds were set to 0.8 and 0.75, respectively.
- If the name similarity of a record pair is greater or equal than its threshold and the similarity of one of the other two attributes (*specialty* or *address*) also satisfies its respective threshold, the records are considered related, regardless of the similarity score of the third attribute.

Based on the above assumptions, Algorithm 2 below describes our implementation for the matcher combination.

Algorithm 2 Algorithm for matcher combination.

```
if relevance(name1) ≥ 0.2 and relevance(name2) ≥ 0.2 then
  if nameSimilarity > 0.8 and (specialtySimilarity > 0.8 or addressSimilarity > 0.75)
  then
    MATCH
else
  if nameSimilarity > 0.9 and (specialtySimilarity > 0.8 or addressSimilarity > 0.75)
  then
    MATCH
```

4.2 Experimental Results

In this subsection, we present a characterization of the records classified as matches and assess the accuracy of our method by sampling the results. In this execution of our method, with the thresholds $B = 5$ and $S = 0.6$, about 119.5 million comparisons have been performed, resulting in a total of 799,877 matched pairs, whereas approximately 118.6 million pairs have been classified as not matched.

Table 2 shows the percentage of pairs found by our method that present exact match for each attribute combination, i.e., they would be matched if an exact match approach were employed. For example, 21.5% of the pairs matched by our method present identical names and addresses, so they would be also matched by an exact match solution that considered only the attributes *name* and *address*. Moreover, only 2.1% of the related records have presented an exact match and more than 16,500 records (2.1%) do not have even the same name. Thus, the exact match approach would not be a good solution as many record pairs that have been matched (and are likely to be related) would not be matched.

Table 2. *Percentage of pairs of records matched by our algorithm that presents exact match for each set of attributes.*

Percentage of exact matches	97.9	71.3	21.6	69.7	21.5	2.2	2.1
Name	X			X	X		X
Specialty		X		X		X	X
Address			X		X	X	X

As we do not have a labeled dataset, we have sampled the dataset and labeled it manually. Since the number of not matched pairs is huge, it is not feasible to label a representative percentage of them. Thus, we have only labeled those records most likely to be incorrectly classified, i.e., we selected the 300,000 most similar pairs of unmatched records and manually labeled 1% of them. For the set of matched records, we have also selected the 300,000 least similar matched pairs and also manually labeled 1% of them. We have also assumed that the importance order of the attributes is *name*, followed by *specialty* and then by *address*. Thus, the similarity of a pair of records for the sample selection was computed as the combination value of the *name*, *specialty* and *address* similarities weighted by 5, 3 and 1, respectively. We notice that there are some pairs for which we have not been able to decide whether they were related or not. We refer to these pairs as unknown.

Table 3 shows the results evaluation considering the aforementioned sampling of 1% of the records most likely to be incorrectly classified. As we cane see, the results are quite good and, as the false negative rate is just 5%, our future efforts should aim those records that although related are not matched by our method.

Table 3. Results evaluation of the 6000 labeled pairs, 3000 from the matched set and 3000 from the not matched set.

	Real value				
Predicted value	Percentage of	Related	Not related	Unknown	Total
	Matched	95.31	1.43	3.26	100
	Not Matched	5.13	93.43	1.43	100

5 Conclusions

In this paper, we have proposed a simple and effective method for solving the entity matching problem involving data records of healthcare professionals. Our contributions are centered on the three main tasks involved in the matching process: blocking, matcher selection and matcher combination. Our blocking strategy adopts an overlapping mechanism for reducing the number of comparisons between related records. Our matchers are based on existing string matching functions especifically tuned for the problem at hand. Finally, our matcher combination strategy is based on an empirically designed heuristic algorithm. We have evaluated our method by conducting a case study involving data from three Brazilian Web sources of healthcare professionals. Our results show that our method has achieved a true positive rate of over than 95% and a true negative rate of over 93%.

As future work, we aim to improve our matchers by hierarchically labeling the medical specialties and splitting addresses into components such as street, number, city and state. We also aim to investigate a graph-based approach as an alternative to address the entity matching problem in the medical domain. Furthermore, we want to investigate the problem of data fusion, i.e., as in many situations the data related to a same entity might also present some conflict, it is very important to apply a method for deciding which information is correct and updated. Finally, we plan to compare our method with other existing entity matching approaches.

Acknowledgements. This work is partially funded by InWeb (grant MCT-CNPq 573871/2008-6), and by the authors' individual scholarships and grants from CAPES, CNPq and FAPEMIG.

References

1. Agrawal, R., Ieong, S.: Aggregating Web Offers to Determine Product Prices. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. pp. 435–443 (2012)
2. Arasu, A., Ganti, V., Kaushik, R.: Efficient Exact Set-similarity Joins. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. pp. 918–929 (2006)
3. Brizan, D.G., Tansel, A.U.: A Survey of Entity Resolution and Record Linkage Methodologies. Communications of the IIMA 6(3), 41–50 (2006)

4. de Carvalho, M.G., Laender, A.H.F., Gonçalves, M.A., da Silva, A.S.: A Genetic Programming Approach to Record Deduplication. IEEE Trans. Knowl. Data Eng. 24(3), 399 – 412 (2012)
5. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer-Verlag (2012)
6. Cohen, W., Ravikumar, P., Fienberg, S.: A Comparison of String Metrics for Matching Names and Records. In: Proc. of the KDD Workshop on Data Cleaning and Object Consolidation. vol. 3, pp. 73–78 (2003)
7. Draisbach, U., Naumann, F.: A Generalization of Blocking and Windowing Algorithms for Duplicate Detection. In: Proc. of the Int'l Conf. on Data and Knowledge Engineering. pp. 18–24 (2011)
8. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate Record Detection: A Survey. IEEE Trans. on Knowl. and Data Engineering 19(1), 1–16 (2007)
9. Ferreira, A.A., Gonçalves, M.A., Laender, A.H.F.: A Brief Survey of Automatic Methods for Author Name Disambiguation. SIGMOD Record 41(2), 15 – 26 (2012)
10. Hassanzadeh, O., Chiang, F., Lee, H.C., Miller, R.J.: Framework for Evaluating Clustering Algorithms in Duplicate Detection. Proc. of VLDB Endow. 2(1), 1282–1293 (2009)
11. Hernández, M.A., Stolfo, S.J.: The Merge/Purge Problem for Large Databases. SIGMOD Record 24(2), 127–138 (May 1995)
12. Kang, I.S., Na, S.H., Lee, S., Jung, H., Kim, P., Sung, W.K., Lee, J.H.: On co-authorship for author disambiguation. Information Processing and Management 45(1), 84 – 97 (2009)
13. Kolb, L., Thor, A., Rahm, E.: Dedoop: Efficient Deduplication with Hadoop. Proc. of VLDB Endow. 5(12), 545 – 550 (2012)
14. Köpcke, H., Thor, A., Thomas, S., Rahm, E.: Tailoring Entity Resolution for Matching Product Offers. In: Proc. of the 15th Int'l Conf. on Extending Database Technology. pp. 545–550 (2012)
15. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data and Knowledge Engineering 69(2), 197 – 210 (2010)
16. Lee, M.L., Ling, T.W., Low, W.L.: IntelliClean: A Knowledge-based Intelligent Data Cleaner. In: Proc. of the Sixth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. pp. 290–294 (2000)
17. McCallum, A., Nigam, K., Ungar, L.H.: Efficient Clustering of High-dimensional Data Sets with Application to Reference Matching. In: Proc. of the Sixth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. pp. 169–178 (2000)
18. Oliveira, J.W., Laender, A.H., Gonçalves, M.A.: Remoção de Ambiguidades na Identificação de Autoria de Objetos Bibliográficos. In: Anais do XX Simpósio Brasileiro de Banco de Dados. pp. 205–219 (2005), (In Portuguese)
19. Raad, E., Chbeir, R., Dipanda, A.: User Profile Matching in Social Networks. In: Proc. of the 13th Int'l Conf. on Network-Based Information Systems. pp. 297–304 (2010)
20. Singla, P., Domingos, P.: Entity Resolution with Markov Logic. In: Proc. of the Sixth Int'l Conf. on Data Mining. pp. 572–582 (2006)
21. Tan, Y.F., Kan, M.Y., Lee, D.: Search Engine Driven Author Disambiguation. In: Proc. of the 6th ACM/IEEE-CS Joint Conf. on Digital Libraries. pp. 314–315 (2006)
22. Thor, A., Rahm, E.: MOMA - A Mapping-based Object Matching System. In: Proc. of the Third Biennial Conference on Innovative Data Systems Research (2007)