Automatic Identification of Best Attributes for Indexing in Data Deduplication

Levy Souza, Fabricio Murai, Ana Paula C. da Silva and Mirella M. Moro

Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
{levysouza, murai, ana.coutosilva, mirella}@dcc.ufmg.br

Abstract. We introduce an approach that selects relevant attributes to the indexing step of data deduplication, reducing the whole processing time and improving the deduplication effectiveness. We evaluate the proposed method on synthetic and real datasets over distinct domains. We also evaluate the impact of choosing the indexing attributes over the other steps of the deduplication process, then concluding our solution is both efficient (time cost) and effective (results quality) as a whole.

Keywords: Duplicate Detection, Deduplication, Indexing

1 Introduction

Data deduplication identifies and eliminates duplicate records from datasets and databases. Duplicate records are data instances that represent the same object in the real world. Data deduplication has several applications. For example, it helps identifying duplicate products in online stores [13], duplicate professionals in healthcare [3] and duplicate contacts in mobile devices [1]. On broader contexts, in information retrieval, it is important for removing duplicate documents (e.g., web pages and bibliographic citations) from the results returned by search engines, digital libraries and automatic text indexing systems [2,7,10,14].

The process of identifying and eliminating duplicate records is composed by three steps: indexing (defines a key per record), record comparison (groups records according to key values), and classification (compares records within the groups). Our focus is on the **indexing** step, which creates *block keys* (BKs) structures to group similar values [5]. Indexing is paramount because it avoids the need to compare all $(n \times (n-1))/2$ record pairs for a source with n records.

Existing approaches use BKs for different purposes, such as sorting and clustering. Examples include standard blocking [8], sorted neighborhood [9], canopy [11], and adaptive sorted neighborhood algorithms [16]. Nonetheless, in all these cases, BK values are defined based on the available attributes. Ideally, such indexing attributes should be *effective* (as to better distinguish the values) and *efficient* (as to allow faster deduplication runtime). For instance, the time to deduplicate a synthetic dataset with a million records ranges from 15 hours to more than a day, depending on the indexing attribute (Section 5). Next, we present a motivating example to better justify our work followed by a summary of current solutions and our contributions.

Table 1: Identifying duplicate records from two data sources

CD-ID	ID	Source	Title	Artist	Category	Genre	Extra	Year
A	1	Atlantic Records	The Moment	Kenny G	Jazz	Jazz	ID3G: 8	1996
	2	J Records	Moment	Kenny G	Null	Classical	Null	1996
B	3	Atlantic Records	Donde Hay Musica	Erros Ramazzotti	Jazz	Ballad	Null	Null
C	4	Atlantic Records	Alien Ant Farm	Anthology	Rock	Rock	Null	Null
	5	J Records	Alien Ant Farm	Anthology	NewAge	AlternRock	ID3G: 40	2001

Motivating Example. Table 1 presents a toy example with five data instances that come from two sources and are identified by the *ID* column. Such instances refer to three music CDs identified by the *CD-ID* column. The goal is to identify and remove the duplicate records from these two sources. At indexing step, records are indexed by an attribute to avoid comparing all 10 pairs of records. Then, depending on the selected indexing attribute, the following could occur: (*i*) indexing by *Source* defines one block for the records of *Atlantic Records* and another for *J Records* – CDs 1 and 2 are not compared against each other, then jeopardizing effectiveness; (*ii*) indexing by *Year* defines two blocks with CDs {1,2} and {5} – the process then compares CDs 1 and 2, but fails to compare 4 and 5; (*iii*) indexing by *Category*, *Genre* and *Extra* is equally ineffective due to the same reasons, and different sources may have distinct interpretations for category and genre; (*iv*) indexing by *Artist* or *Title* may potentially result in more effectiveness, as it compares CDs 1 and 2, as well as 4 and 5.

Overview of Current Solutions. Initially, the indexing attribute selection process was purely arbitrary [9]. Nowadays, it is based on expert knowledge about the data domain [6,12]. There are two main strategies that use available attributes to define BKs: *schema-agnostic*, which uses the entire values of attributes; and *schema-based* configurations, which combine rules of values extracted from each attribute [12]. In either case, experts are required to choose the best attributes. Also, most studies focus on selecting the best attributes for the *classification* step [2,4,13]. In that step, the goal is to choose the best attributes to compare the values. Current solutions include machine learning approaches [4] and strategies using data information [2]. Nonetheless, there is no solution tailored for the initial indexing step.

Our Contributions. We propose a new method for indexing-oriented attribute selection and assess its performance through an extensive experimental evaluation. We measure the efficiency and effectiveness of the proposed strategy in real and synthetic datasets. The datasets cover a wide spectrum of data domains including bibliographic records, music CDs, personal information and restaurant data. Results show that using the best-ranked attribute by the proposed method yields the best F-Measure results on 10 out of 13 datasets. The deduplication outcome improves by about 88% in terms of F-Measure when using the best attribute instead of the worst one. Finally, the implementations of the methods and the datasets are publicly available[1].

[1] Sources: `http://www.dcc.ufmg.br/~mirella/projs/deduplica`

2 Data Deduplication Process

Let D be a dataset containing i records, such that $D = \{r_1, r_2, .., r_i\}$. Each record r is defined by a set of attributes $A_r = \{a_1, a_2, .., a_j\}$. Then, the data deduplication process is composed of three steps: indexing, record comparison and classification, each generating input to the following one. Next, we describe such steps and the most commonly used algorithms to process each one [5].

Indexing Step. Here, the goal is to assign a block key value to each record. First, one attribute is chosen from the available set. Next, the BK value is set to be the attribute value or an encoding applied to the attribute value. After indexing, each $r \in D$ is associated with a BK value. One popular technique is Soundex [5]: it encodes attribute values by keeping the first character and converting the others to digits between 0 and 6. For instance, the values *Rock*, *Jazz* and *Classical* (Table 1) are encoded as $R200$, $J200$ and $C422$.

Record Comparison Step. After the indexing step, the records are grouped based on BK values using (one of) two main approaches.

The *Standard Blocking Algorithm* creates a set of blocks where each block groups similar records. Hence, records within a block are compared only against each other (i.e, reducing the original quadratic complexity). These groupings are defined according to the BK values generated for each record. For instance, in Table 1, when indexing records 1, 2, 3, 4 and 5 through the *Artist* column and using the Soundex encoding, three groups are created: $K520$ with records 1 and 2; $E626$ with 3; and $A534$ with records 4 and 5.

The *Sorted Neighborhood Algorithm* [9] combines the records through a *sorting key*, which is similar to a BK. However, before performing the comparisons, all records are sorted according to their BK values. Then, a sliding window of size $w > 1$ traverses all records of D, and the first record of the window is compared to all others in the same window.

Classification Step. For evaluating how similar two values are, a similarity function usually calculates the correspondence between them and returns a number in $[0, 1]$, where 1 means "perfect match". The *Jaro Winkler algorithm* (a popular extension of Jaro algorithm proposed by Winkler [15]) considers the size of strings and the types of errors that commonly occur with alphanumeric variables to calculate the similarity. At the end of this step, the records are classified as *match*, *non-match* and *possible match* based on a similarity threshold applied to all attributes values (e.g., *Source*, *Title*, *Artist*, *Category*, *Genre*, *Extra*, *Year*) or just to the most descriptive attributes of D (e.g.,*Title* and *Artist*) [2].

3 Identification of Best Attributes for Indexing

We now introduce our method for selecting the best indexing attribute. It creates a ranking for each attribute $a \in A_r$ based on a combination of metrics described ahead. It does not use machine learning algorithms, being adaptable to any data domain without needing a training dataset. It also works without requiring intervention of experts, huge advantage over existing approaches.

Metrics Definition. Our proposed method is based on four metrics, computed for each attribute $a \in A_r$: (*1*) *Duplicity* is the ratio between number of duplicate values and number of not-null instances; (*2*) Distinctiveness is the ratio between number of distinct values and number of not-null instances; (*3*) *Density* is the fraction of non-null instances; and (*4*) *Repetition* is the ratio between number of repeated records and number of distinct values, defined as follows.

$$\overline{\text{Dup}}(a) = \frac{\text{dupValues}(a)}{\text{notNull}(a)} \quad (1) \qquad \overline{\text{Dist}}(a) = \frac{\text{distValues}(a)}{\text{notNull}(a)} \quad (2)$$

$$\overline{\text{Dens}}(a) = \frac{\text{notNull}(a)}{T} \quad (3) \qquad \overline{\text{Rep}}(a) = \frac{T - \text{distValues}(a)}{\text{distValues}(a)} \quad (4)$$

where notNull(a) is the number of valid, non-null instances for an attribute a, dupValues(a) is the number of duplicate values for a, distValues(a) is the total of distinct values for a, and T is the total of instances. We normalize each metric by its maximum over A_r and denote them by Dup(a), Dist(a), Dens(a) and Rep(a).

One advantage is that our metrics can be easily computed over any relational dataset, being commonly present in the histogram features of relational systems. Hence, the proposed method can be adaptable for several data domains. Also, density and repetition are used by others to select relevant attributes for the classification step [2] (but this is the first time for the indexing step).

Attribute Relevance Calculation. Our solution defines a relevance score for each attribute based on the metric values. The unnormalized relevance score of attribute a is defined as

$$\bar{R}(a) = \text{Dens}(a) + \text{Dup}(a) + ((1 - \text{Dist}(a)) \times \text{Dens}(a)) + (1 - \text{Rep}(a)). \quad (5)$$

Intuitively, density and duplicity improve efficacy. High repetition and distinct values lead to large run times because they create few blocks with many records (i.e., many comparisons) or many blocks with few records (i.e., several accesses to the database to retrieve the records of each distinct BK). Hence, we consider their complement to improve efficiency, as attributes without much repetition and distinctiveness perform faster. Also, we multiply $(1 - \text{Dist}(a))$ by Dens(a) to avoid that low distinctiveness and density have high scores. Last, we normalize $\bar{R}(a)$ by its maximum value to obtain the attribute relevance score $R(a)$.

Example. For the example in Table 1, we compute the metrics for each attribute $a \in \{$*Source*, *Title*, *Artist*, *Category*, *Genre*, *Extra*, *Year*$\}$ and calculate its relevance using Equation 5. Next, we sort each attribute $a \in A_r$ by its relevance $R(a)$. Table 2 summarizes the resulting metric values and relevance of each attribute, making *Artist* the best attribute for the indexing step. Hence, we expect that using *Artist* in the indexing step will yield the best results (w.r.t. efficiency and efficacy) in data deduplication; whereas *Extra* will yield the worst ones (confirmed later on Figure 3(a) in Section 5).

4 Experimental Setup

Experimental Methodology. We implemented the data deduplication algorithms from Section 2 in Java and performed the experiments on a Intel *i*7

Table 2: Best attributes for indexing

Attribute	**Dens**	**Dup**	**Dist**	**Rep**	$\overline{\mathbf{R}}(a)$	$\mathbf{R}(a)$
Artist	1.00	0.67	0.60	0.44	2.62	1.00
Source	1.00	1.00	0.40	1.00	2.60	0.99
Title	1.00	0.33	0.80	0.17	2.37	0.90
Category	1.00	0.33	0.80	0.17	2.37	0.90
Year	0.60	0.56	0.67	0.33	2.02	0.77
Genre	1.00	0.00	1.00	0.00	2.00	0.76
Extra	0.40	0.00	1.00	0.00	1.40	0.53

Table 3: Experimental datasets

Datasets				**Replications**	
#	**Dataset**	**#Inst.**	**#Dup.**	**#Efficiency**	**#Efficacy**
1	Cora	1,879	1,688	50	1
2	Cds	9,763	222	50	1
3	Restaurant	864	112	50	1
4	Dup10%	11,000	1,000	30	10
5	Dup30%	13,000	3,000	30	10
6	Dup50%	15,000	5,000	30	10
7	Dup70%	17,000	7,000	30	10
8	Dup90%	19,000	9,000	30	10
9	DB10^2	110	10	100	5
10	DB10^3	1,100	100	30	5
11	DB10^4	11,000	1,000	30	5
12	DB10^5	110,000	10,000	30	5
13	DB10^6	1,100,000	100,000	10	5

(2.3 GHz) desktop computer with 16GB of RAM, running MAC OS X 10.11.3. The experimental methodology consists in performing the complete data deduplication process with different indexing setups for each attribute available in the dataset. For the other steps, we use standard blocking or the sorted neighborhood algorithm for record comparison, and the Jaro Winkler algorithm for classification. In the classification step, we compute the similarity among the most descriptive attributes of each dataset. Then, we compute efficiency and effectiveness for each attribute. Specifically, at efficiency, we compute the time to retrieve the distinct BK values and perform the data deduplication process and, in effectiveness, we consider the F-Measure. Our goal is to evaluate whether the most efficient and effective data deduplication results are achieved using the best ranked attributes by the proposed method. We use 95% confidence interval and 5% of maximum error to define replications as shown in Table 3.

Datasets. We use synthetic and real datasets with a variety of contexts and attribute types. The synthetic datasets are created by the Data Set Generator Program [5]. It creates original and duplicate records according to the parameters set by the user. Each record has the following attributes: names, addresses, dates, phone numbers, and identifier numbers. In particular, our evaluation considers eight attributes: *Given Name*, *Surname*, *Address1*, *Address2*, *Suburb*, *Culture*, *State* and *Title*. To generate the synthetic datasets, the input parameters are set to the same values used in previous work [2] that uses this software: maximum amount of duplicates per record = 3, maximum amount of changes per field = 5, maximum amount of changes per instance = 5, probability distribution is set to "uniform", type of change is set to "all", and number of households = 1. The number of original and duplicate records per dataset is in Table 3.

We also use three real datasets available from the DuDe toolkit[2]: CORA, Restaurant and CD Information. These are widely used on data deduplication research, e.g., [2] and [6]. CORA consists of bibliographical information about scientific papers, providing 1,879 instances. Restaurant data is a collection of 864 restaurant records from the Fodor's and Zagat's restaurant guides and contains

[2] Duplicate Detection (DuDe) toolkit: `https://hpi.de/naumann/projects/data-quality-and-cleansing/dude-duplicate-detection.html`

112 duplicates. CD information includes 9,763 CDs randomly extracted from freeDB. Table 3 summarizes all datasets statistics.

Parameter Tuning and Evaluation Metrics. Regarding parameter tuning, we empirically evaluated different setups to choose the values that provide the best results. Overall, for Jaro Winkler the similarity threshold is 0.9, and for sorted neighborhood the window sizes vary from 1% to 5% depending on the dataset size. We compute the commonly used Precision, Recall and F-Measure to evaluate the data deduplication process, attribute effectiveness and the proposed method [5]. For conciseness, we only show the F-Measure results as it considers a harmonic mean of Precision and Recall.

5 Experimental Results

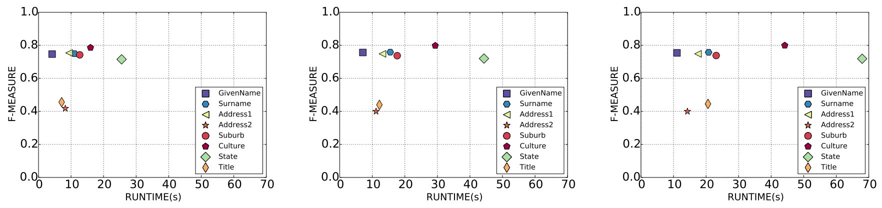
In this section we present experimental results on the performance of the proposed method on several domains of synthetic datasets and on the three real datasets. The deduplication results are described in terms of F-Measure and runtime. Let $F(a_i)$ be the F-measure yield by attribute $a_i \in A_r$ and $F_{\max} = \max_{a_i \in A_r} F(a_i)$. We consider as candidates to best attribute the set $B = \{a_j \in A_r : F_{\max} - F(a_j) < 0.1\}$. The **best attribute** is defined as the candidate in B that has the smallest runtime. With this approach, we can guarantee efficiency without jeopardizing effectiveness. Due to space constrains, only the results in *Standard Blocking* algorithm are shown.

Synthetic Datasets Evaluation. This experimental evaluation with synthetic data aims to verify if: (i) the proposed method selects efficient and effective attributes in all datasets for various amounts of duplicate records; and (ii) the proposed method scales well for large datasets.

First, we measure the performance of the data deduplication process by using the standard blocking algorithm and varying the amount of duplicate records in the dataset (10%, 30%, 50%, 70% and 90% of duplicate records). Figure 1 presents the results in terms of F-measure (Y-axis) and runtime (X-axis) obtained by each possible choice of attribute in A_r for the indexing step. We only show the results in the datasets with 10%, 50% and 90% of duplicate records because the outcomes are similar in the others (i.e., 30% and 70%). The best data deduplication results are located on attributes near the top left of each chart (faster and more effective attributes). In this case, the performance of each attribute was consistent across datasets.

Next, we calculate the metrics defined in Section 2 for each attribute in each dataset. Then, Table 4 shows the normalized attribute relevance (Equation (5)) combining such metrics. We note that the proposed method selects the best attributes for all datasets, i.e., the most relevant attributes (largest $R(a)$) are those with the best data deduplication results (Figure 1). Similar to the attributes' performance, the ranking obtained by $R(a)$ was also consistent across datasets.

In all datasets, the best attribute is the *Given Name*. Specifically, in the dataset with 10% of duplicate records, *Given Name* has an F-Measure of $0.7467\pm$ (0.0101) and runtime of $4.184 \pm (0.0350)$ seconds, and is almost two times more



(a) 10% of Duplicates (b) 50% of Duplicates (c) 90% of Duplicates

Fig. 1: Effectiveness vs. efficiency by indexing attribute - #duplicates evaluation

Table 4: Amount of duplicate records for selecting of best attributes

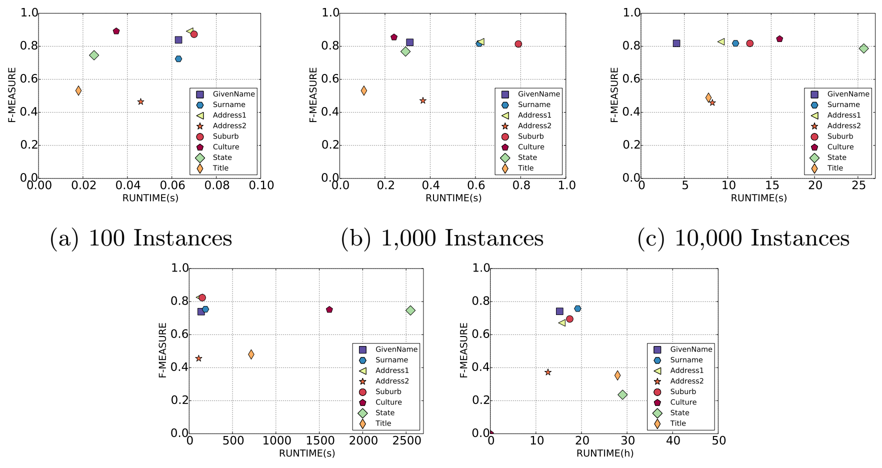
10% Duplicates		**50% Duplicates**		**90% Duplicates**	
Attribute	R(a)	Attribute	R(a)	Attribute	R(a)
GivenName	1.0000	GivenName	1.0000	GivenName	1.0000
Address1	0.9553	Address1	0.9515	Surname	0.9522
Suburb	0.9269	Suburb	0.9460	Suburb	0.9520
Surname	0.9184	Surname	0.9443	Address1	0.9470
State	0.8118	State	0.9272	State	0.9452
Culture	0.6709	Culture	0.6816	Culture	0.6883
Address2	0.5670	Address2	0.5842	Address2	0.5920
Title	0.3037	Title	0.4345	Title	0.5007

effective than *Address2* and six times faster than *State*. In addition, the worst attributes (*Title* and *Address2*), i.e., fast but ineffective, are the least relevant attributes according to the proposed method for all datasets. These results indicate that the proposed solution selects efficient and effective attributes for datasets with various amount of duplicate data.

To address the second point, we evaluate the proposed method when using the standard blocking algorithm and varying the total number of instances in $\{10^2 - 10^6\}$ instances. Figure 2 presents the performance results obtained by each possible choice of attribute in A_r for the indexing step. Note that we do not fix the X-axis scale because the goal is to pinpoint the best attributes for each dataset, instead of comparing the runtimes across datasets. Table 5 shows the attribute relevance ranking for each dataset.

We note that the proposed method selects the best attributes for all datasets except for those with 10^3 and 10^6 instances. In those datasets, the second most relevant attribute has the best data deduplication results. In addition, the worst attributes (*Title* and *Address2*) are ranked as the least relevant by the proposed method in all datasets. *Address2* is fast but ineffective, whereas *Title* is ineffective in all datasets. In the dataset containing 10^6 instances, there are attributes that yield runtimes longer than one day (*State*), while others take approximately 15 hours. This highlights the importance of selecting the best attributes for the indexing step. These results indicate that the proposed method is efficient and effective for datasets of very different scales.

Real Datasets Evaluation. We now evaluate the proposed method in three real datasets from different domains – Cora, CDs and the Restaurant datasets,



(a) 100 Instances (b) 1,000 Instances (c) 10,000 Instances

(d) 100,000 Instances (e) 1,000,000 Instances

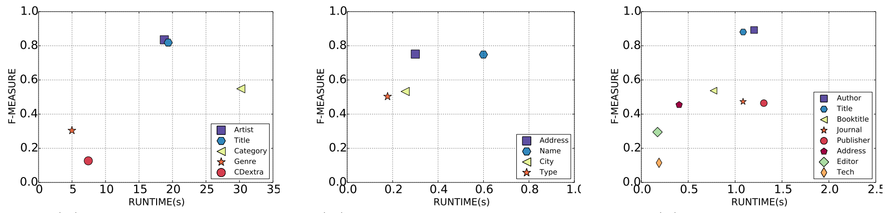
Fig. 2: Effectiveness vs. efficiency by indexing attribute - #instances evaluation

Table 5: Total of instances for selecting of best attributes

10^2 **Instances**		10^3 **Instances**		10^4 **Instances**		10^5 **Instances**		10^6 **Instances**	
Attribute	R(a)	Attribute	R(a)	Attribute	R(a)	Attribute	R(a)	Attribute	R(a)
Culture	1.0000	GivenName	1.0000	GivenName	1.0000	Suburb	1.0000	Suburb	1.0000
GivenName	0.9393	Culture	0.9035	Address1	0.9632	GivenName	0.9945	GivenName	0.9976
Address1	0.8387	Address1	0.8510	Suburb	0.9333	Address1	0.9785	Surname	0.9902
Surname	0.8351	Surname	0.8418	Surname	0.9217	Surname	0.9778	Address1	0.9806
Suburb	0.8323	Suburb	0.8146	State	0.8074	State	0.9040	State	0.9348
State	0.7518	State	0.6571	Culture	0.6707	Culture	0.6588	Culture	0.6563
Address2	0.5983	Address2	0.5520	Address2	0.5723	Address2	0.6039	Address2	0.6049
Title	0.5090	Title	0.3513	Title	0.2831	Title	0.4668	Title	0.5394

all of which are widely used in data deduplication experiments. We investigate whether the proposed method selects efficient and effective attributes for real datasets from different domains. Figure 3 presents the data deduplication results in terms of F-Measure and runtime, and Table 6 shows the attributes ranking in the corresponding datasets. We use the standard blocking algorithm for the record comparison step.

We observe that the proposed method also identifies the best performing attributes in the real datasets. For the CDs dataset, the most relevant attribute is *Artist* (Table 6), which has the best data deduplication result (Figure 3a). For the Restaurant dataset, the proposed method does not identify the best attribute. However, the second most relevant attribute (*Address1*) has significant efficiency and effectiveness in the data deduplication process. For CORA, the most relevant attributes (*Author* and Title) have the best results in deduplication. Overall, our results indicate that the best performing attributes in real datasets tend to be ranked first or at least high, regardless of the data domain. Hence, it can



(a) CDs Dataset (b) Restaurant Dataset (c) Cora Dataset

Fig. 3: Effectiveness vs. efficiency by indexing attribute - real datasets evaluation

Table 6: Selection of best attributes - real datasets

CDs Dataset		**Restaurant Dataset**		**Cora Dataset**	
Attribute	R(a)	Attribute	R(a)	Attribute	R(a)
Artist	1.0000	Type	1.0000	Author	1.0000
Genre	0.9711	Address	0.9123	Title	0.9567
Title	0.8884	Name	0.9120	BookTitle	0.6505
Category	0.8594	City	0.8207	Address	0.5706
DExtra	0.6491			Editor	0.4922
				Publisher	0.4798
				Tech	0.3668
				Journal	0.2507

be applied to other relational datasets, being advantageous relative to machine learning algorithms, since it does not require training data.

Results Discussion. This evaluation showed the indexing attribute has a large impact on the efficiency and effectiveness of data deduplication. Regarding effectiveness, in the 10% of duplicate records dataset, attributes *Culture* and *Address2* differ by about 88% (F-Measure of 0.7857 vs. 0.4179). Furthermore, there may be at least four attribute scenarios in the data deduplication results: efficient and effective attribute (*Given Name*); efficient and ineffective attributes (*Title* and *Address2*); inefficient attribute (State); and effective but less efficient attributes (*Address1*, *Surname*, *Suburb* and *Culture*). Regarding runtime, using inefficient attributes for deduplication can take more than one day to run for large datasets, while using more efficient attributes allows it finish in about 15 hours. Thus, selecting the best attribute for indexing is a crucial step.

6 Concluding Remarks

In this work, we presented a method for automatically selecting the best attributes for the first step on data deduplication processing: indexing. The goal was to rank attributes, enabling to identify those that would provide more efficient and effective deduplication. In the experimental evaluation, we evaluated the proposed method on synthetic and on real datasets. Moreover, we assessed questions related to: the efficiency and effectiveness of the indexing attribute considering two of the main methods for record comparison (blocking and neighbor); the indexing functions (agnostic and configurations-based), and the combination of indexing attributes. Due to space constraints, we only showed the results in standard blocking algorithm. Finally, we also verified the effectiveness of the proposed method for selecting relevant attributes.

Our main conclusion is the most efficient and effective data deduplication results are achieved by using the best ranked attributes as given by our method. We are in the process of comparing the proposed method against baselines. In the future, we plan to evaluate our solution over other datasets, analyze cases of failures in which it does not identify the best attribute (Restaurant Dataset) as well as evaluate metrics of false positives and negatives.

Acknowledgments. Research partially funded by CNPq and FAPEMIG, Brazil. The authors also thank PPGCC/UFMG for financial support.

References

1. Borges, E.N., et al.: Contact deduplication in mobile devices using textual similarity and machine learning. In: ICEIS. pp. 64–72 (2017)
2. Canalle, G.K., et al.: A strategy for selecting relevant attributes for entity resolution in data integration systems. In: ICEIS. pp. 80–88 (2017)
3. Carvalho, L.F.M., et al.: Entity matching: A case study in the medical domain. In: AMW (2015)
4. Chen, J., et al.: A learning method for entity matching. In: QDB (2012)
5. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Berlin (2012)
6. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. TKDE 24(9), 1537–1555 (2012)
7. Davis, P., et al.: Methods for precise named entity matching in digital collections. In: JCDL. pp. 125–127 (2003)
8. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
9. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: ACM SIGMOD. pp. 127–138 (1995)
10. Levin, F.H., Heuser, C.A.: Using genetic programming to evaluate the impact of social network analysis in author name disambiguation. In: AMW (2010)
11. McCallum, A., et al.: Efficient clustering of high-dimensional data sets with application to reference matching. In: ACM SIGKDD (2000)
12. Papadakis, G., et al.: Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. PVLDB 9(4), 312–323 (2015)
13. Su, W., et al.: Record matching over query results from multiple web databases. TKDE 22(4), 578–589 (2010)
14. Vieira, P., et al.: A query-driven, incremental process for entity resolution. In: AMW (2016)
15. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research. pp. 354–359 (1990)
16. Yan, S., et al.: Adaptive sorted neighborhood methods for efficient record linkage. In: JCDL. pp. 185–194 (2007)