Applying term frequency-based indexing to improve scalability and accuracy of probabilistic data linkage

Robespierre Pita[1,2], **Luan Menezes**[1,2], **Marcos E. Barreto**[1,2]

[1]Institute of Mathematics and Statistics, Computer Science Department, Federal University of Bahia (UFBA), 40.170-110, Salvador, BA, Brazil

[2]Centre for Data and Knowledge Integration for Health (CIDACS), Oswaldo Cruz Foundation (FIOCRUZ), 41.940-220, Salvador, BA, Brazil

[robespierredrp, luanmenezes, marcoseb]@dcc.ufba.br

Abstract. *Record or data linkage is a technique frequently used in diverse domains to aggregate data stored in different sources that presumably pertain to the same real world entity. Deterministic (key-based) or probabilistic (rule-based) linkage methods can be used to implement data linkage, being the second approach suitable when no common link attributes exist amongst the data sources involved. Depending on the volume of data being linked, indexing (or blocking) techniques should be used to reduce the number of pairwise comparisons that need to be executed to decide if a given pair of records match or not. In this paper, we discuss a new indexing scheme, based on term-frequency counts, deployed in our data linkage tool (AtyImo). We present our algorithm design and some metrics related to accuracy and efficiency (reduction ratio achieved during blocking construction), as well a comparative analysis with a predicate-based technique also used in AtyImo. Our results shows a very high level of accuracy and reduction in terms of pairwise comparison tasks.*

1. Introduction

Record linkage is a well-known technique used to integrate data by finding records potentially belonging to the same entity on distinct sources [Newcombe et al. 1959]. It is considered an important approach in several domains as the data aggregated from different data sources can be used to build a complete view of a given scenario, as well support diverse types of studies and decision-making processes.

In Public Health, this technique is frequently used to aggregate data from different electronic health records (EHR) usually managed by governmental bodies within the public health system. Throughout his life, an individual (or patient) has access to a diversity of EHR systems that store specific data on clinical care episodes, symptoms and diseases, prescribed medication and treatments, and associated outcomes. Being able to aggregate these data is a crucial step to build the patient's history, as well support different types of studies, such as quasi-experimental analysis, clinical trials and longitudinal (cohort-based) evaluations [Newcombe et al. 1959]. The Brazilian Public Health System (SUS) is comprised by dozens of freely accessible databases providing anonymized data on live births, mortality, notifiable diseases, nutritional growth, hospital episodes etc. These databases present a high degree of structural heterogeneity and coverage periods although being managed, most of them, by a central department (DATASUS). Heterogeneity is related to the lack of common key attributes amongst all databases that hinder the usage of

a deterministic approach to integrate data pertaining to the same individual. In such situations, we should rely on probabilistic approaches to retrieve as much records as possible, as well on effective ways to validate the retrieved records as truly positive pairs (i.e. make sure recovered pairs actually belong to the same individual).

The scope of our work comprises the usage of term frequency-based indexing to improve the accuracy and the scalability of our probabilistic data linkage tool — AtyImo [Pita et al. 2018]. We started designing AtyImo in 2013 as a solution to aggregate data from several Brazilian governmental databases and generate specific data sets for diverse epidemiological studies. These studies are being conducted within joint Brazil-UK projects aiming to build large population-based cohorts and assess the effectiveness of public health policies [1].

AtyImo is a Python-based data linkage tool implemented over Spark and CUDA able to explore highly distributed and hybrid (multicore CPUs + multi-GPU) parallel architectures, respectively. It is used at CIDACS to support data linkage tasks involving a huge population-based cohort (the 100 million cohort) and public health databases, as well the setup of a live birth cohort (around 80 million records) comprised by children diagnosed with microcephaly due to Zika infection. It is also used at UFBA to aggregate data from different sources within the Brazilian malaria ecosystem to support the design and validation of forecasting models applied to malaria epidemics.

AtyImo is structured as a 4-step pipelining implementing data quality assessment, data pre-processing, record linkage (pairwise comparison) and accuracy assessment. The data pre-processing step is responsible for data cleansing and harmonization, block construction and anonymization. Blocking is a mandatory approach to reduce the number of comparisons, specially in scenarios involving huge databases (cohorts) as ours. The basic idea is to define some criteria to group records into blocks and perform comparisons only among blocks presumably similar. Measuring the relative reduction of comparisons may evidence the effectiveness of a blocking solution [Christen and Goiser 2007].

Besides scaling record linkage solutions over huge data sources by reducing comparisons, another issue concerns the capacity of keeping a good accuracy level. To comply with this requisite, a good indexing solution must apply criteria that increase pair completeness, i.e. which records should be included in each block to keep comparisons and increase accuracy [Elfeky et al. 2002].

In this work, we discuss a term frequency-based indexing technique implemented in our Spark-based version of AtyImo. We rely on this technique as an alternative to the existing approach based on predicates. Building a ranking of records with most terms in common can result in fewer and better comparisons than the predicates-based solution. We calculated reduction ratio and accuracy measures to compare the efficiency of our proposal with other indexing algorithms. The results show that term frequency indexing outperforms other existing solutions, decreasing the number of pairwise comparisons and the execution tume, while preserving good accuracy levels.

This paper is structured as follows: Section 2 briefly describe some existing tech-

[1]This work was supported by CNPq, FINEP, FAPESB, Bill and Melinda Gates Foundation (OPP1161996), The Royal Society (NF160879), National Institute for Health Research (RP-PG-040710314) and also supported by the Wellcome Trust (086091/Z/08/Z)

niques used for indexing. Section 3 presents our term-frequency indexing implementation and compare it with the predicates-based method used by the production version of AtyImo. Section 4 discuss some experimental results. Some related works are metioned in Section 5. Finally, section 6 brings some conclusions and current work directions.

2. Indexing techiniques for record linkage

There are several solutions to scale up record linkage by reducing unnecessary comparisons [Christen 2012]. In this work we focus on those already used in AtyImo tool in order to ensure the understanding of our proposal.

2.1. Traditional blocking

Traditional blocking techniques are generally based on a blocking key which is created using attributes like *first name, last name* and *date of birth*, or some combination of them to split the data into buckets. Records that exactly match with the blocking key on each database are compared assuming their potential to be a true link. The reduction ratio of this approach depends on the discriminative power of the attributes used to build the blocking key.

However, this method can be very biased due to the nature of these attributes and imputation errors that can prevent true matches to be compared. To soften this, phonetic codes can be applied to nominal data and some normalization can be done on numerical data. The low complexity and the easy implementation make this strategy very useful in most situations.

2.2. Blocking with predicates

This method is presented as multi-pass sorted neighborhood alternative and consists of building some predicate from fields or portions of them. Finally, a function uses this predicate to make a junction of disjunctions.

To illustrate, we can suppose a patient record including attributes like *name, mother name, date of birth* and *gender*. Some predicates can use the *first name, birth year* and *gender*. A second may contains the *last name, first mother's name* and *date of birth*. So every record that agrees with the expression $(firstname \wedge birthyear \wedge gender) \vee (lastname \wedge mothersfirstname \wedge birthdate)$ will be part of the same block.

The use of predicate-based indexing can prevent input errors to separate true matches from right blocks, thus increasing pair completeness. Other variants can learn the best predicates to use [Bilenko et al. 2006] or cluster nominal values into phonetic codes. However, the best use of this technique relies on choosing very discriminative attributes in order to get smaller and powerful blocks.

3. AtyImo's implementation of term frequency-based indexing

The current version of AtyImo implements indexing through a predicate-based approach. This implementation comprises two predicates: $pr1 = (firstname \wedge (birthday \vee birthmonth \vee birthyear)) \vee (lastname \wedge (birthday \vee birthmonth \vee birthyear))$ and the second as $pr2 = ((firstname \wedge firstmothersname) \wedge (birthday \vee birthmonth \vee birthyear)) \vee ((lastname \wedge lastmothersname) \wedge (birthday \vee birthmonth \vee birthyear))$.

Figura 1. Predicate-based blocking.

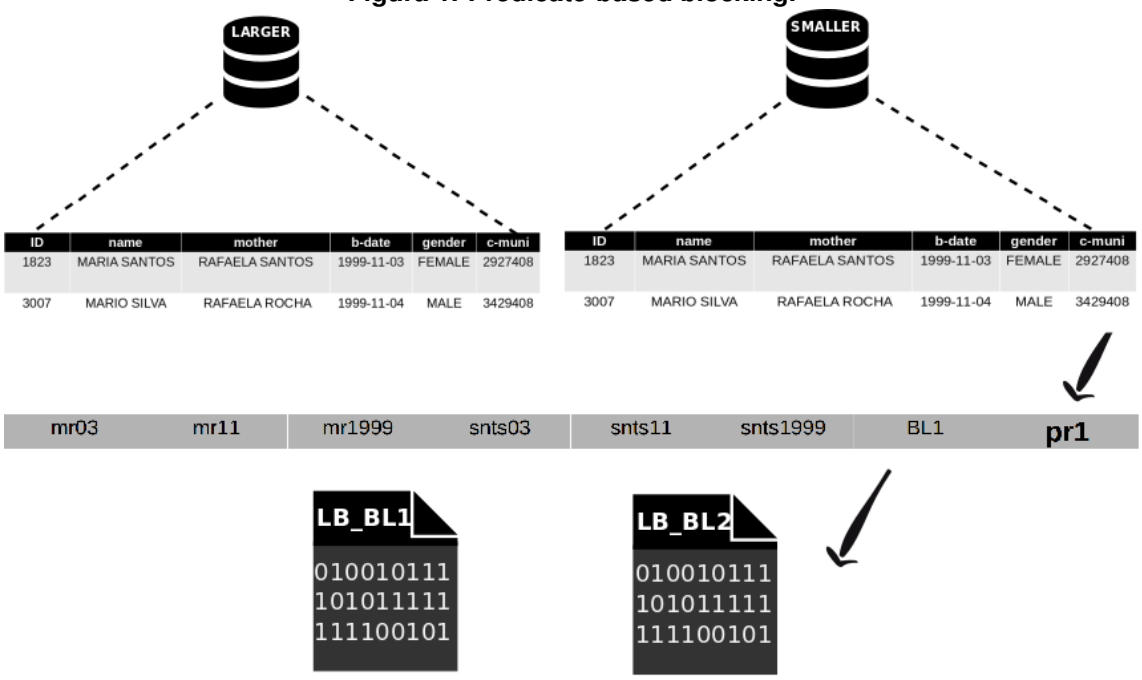
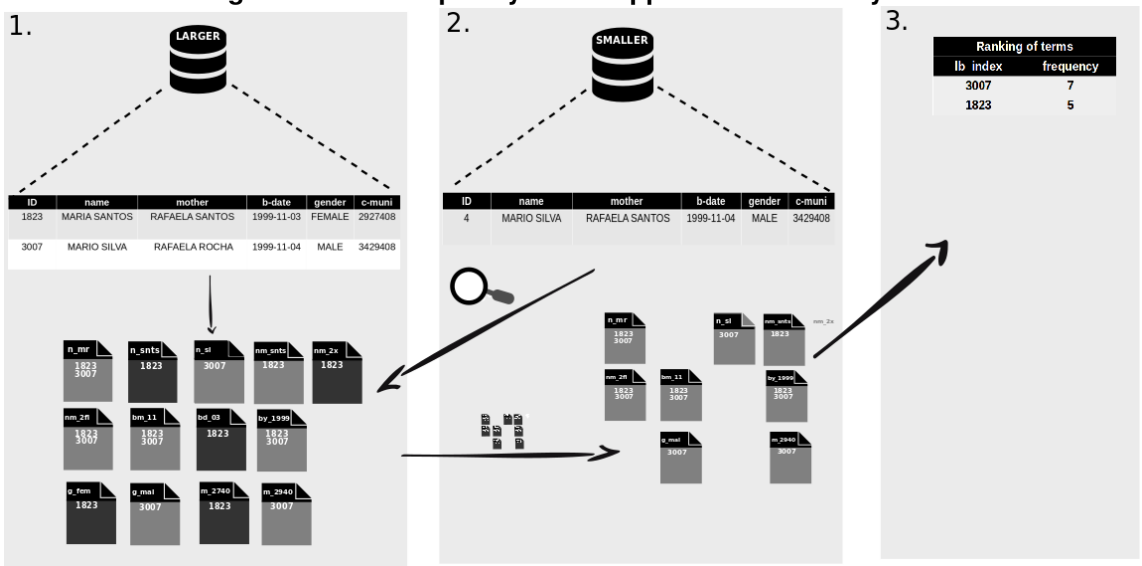


Figure 1 shows how the blocks are build based on predicate $pr1$. The smaller database are used to build an auxiliar file which stores all blocking keys related to each block. After that, a second phase runs over both databases and the auxiliar file to determine in which block a given record must be inserted.

Let consider B_i a database with a finite set o X objects, described by attributes $r = x_1, ..., x_n$. This solution inserted three new steps into AtyImo's pipeline: *indexing, exploration, and ranking* procedures, as illustrated on Figure 2:

Figura 2. Term frequency-based approach used in AtyImo.



3.1. Indexing step

The *indexing step* intend to build a key−value structure by visiting every $X \in B_1$ (larger database) and collecting all terms as keys. The values correspond to every record's primary key (x_1) whose contains the term. To cluster the terms with the same phonetics, we use a custom implementation of Metaphone algorithm [Binstock and Rex 1995] for Brazillian Portuguese. Every term has a prefix to indicate which attribute it came from, this prefix can be n for the name, nm for mother's name, the day, month and year of birth are represented by bd, bm and by, respectively. The prefix g stands for gender and m represents the municipality of birth. Either prefix and phonetic encoded term are set to be the key of the proposed structure and the name of JSON file to store it.

3.2. Exploration step

To every term extracted from an $X \in B_2$ (smaller database this time), we search the respective JSON file and concatenate their values into a vector of indexes vt. Very popular terms can make this phase too onerous, in order to meet this issue, we submit the processing of these JSON files to a distributed processing supported by pySpark, the Algorithm 1.

3.3. Ranking step

The term frequency is calculated by $\frac{f(t,vt)}{\sum_{t \in vt} f(t',vt)}$. After that, we sort vt to find the most frequent B_1's primary keys to set up a ranking with the best N candidates to be compared to some B_2's record. This approach aims to eliminate the variance of the comparisons amount even to records which contain very popular terms. This work scope is limited to indexing the data, search the best candidates and provide a comparison structure for the further phases of AtyImo (well explained in [Pita et al. 2018].

```
Data: larger dataset
Result: json files indexed
initialization;
while not at end of this
  document do
    read current;
    for current.split(;) to
      l do
        term←metaPTBR(l);

        createJsonFile(term)
    end
end
```

Algorithm 1: indexing step

```
Data: smaller dataset
Result: terms ranking
initialization;
while not at end of this
  document do
    read current;
    for current.split(;) to
      l do
        result←searchJsonFile(l);

        ranking.append(result)
    end
end
mostFrequencyTerms(ranking)
```

Algorithm 2: exploration step

4. Experimental results

To evaluate our term frequency-based implementation, we have used samples from two Brazilian governmental databases: SIM (mortality data, presented as the smaller database) and SINASC (live births data, presented as the larger database), as summarized in Table 1. These data are used as "gold standard" to validate our linkage and deduplication

routines, since they share a common key which refers to a death certificate. We expect to retrieve 3.030 true match pairs among these databases after around 84 million pairwise comparisons. Regarding the known true matches numbers, we understand it is a real-world dataset with some impure data, which limits the identification of all true matches without a specificity and sensitivity trade-off. Without blocking, AtyImo retrieves 3018 true matches. Further discussion on AtyImo's accuracy is found in [Pita et al. 2018].

Tabela 1. Gold standard data set used for validation.

SIM	6,458 records
SINASC	13,046 records
Total of comparisons	84,251,068 records
Expected true positives	3,030 records

We use reduction ratio and pair completeness as the main measures to evaluate the quality of our proposed indexing technique [Christen 2012]. Both measures are presented by the Equations 1 and 2. Where, $BLcs$ is the number of records sets in the block, true and false matchs. On the other hand $BLtm$ is only true matchs. Trs is the total number of sets in the dataset and Ttm is the total of true matches.

$$RR = 1 - \frac{BLcs}{Trs} \qquad (1) \qquad\qquad PC = \frac{BLtm}{Ttm} \qquad (2)$$

We have compared the predicate-based approach used by AtyImo (in its production version) against the new term frequency-based approach proposed in this work. Table 2 presents the distribution of blocks sizes for each indexing method. The $predicate1$ achieve smaller blocks due the discriminative power of attributes used. This result reinforce the need of well choose which portions of data will be submitted to the indexing technique. In spite of predicates steady improvement, the results of term frequency approach outperforms all the other compared solution. Since we have used $N = 100$ to define how much pairwise comparisons will be made to every record on smaller database, the size of blocks became homogeneous.

Tabela 2. Size of generated blocks for each indexing technique

method	**predicate 1**		**predicate 2**		**term frequecy**	
database	***sb***	***lb***	***sb***	***lb***	***sb***	***lb***
min	1	1	1	1	1	100
med	24	51	2	2	1	100
mean	43	88.38	8.289	11.57	1	100
max	1855	41528	87	611	1	100

Table 3 summarizes the results obtained for each method. The $predicate1$ got poor results of reduction ratio by performing almost half of all pairwise comparisons. Even with higher average block size, $predicate1$ execution only retrieve 2,382 from the 3,030 expected, which decrease it pair completeness to 0.786. Better results has been made by $predicate2$, which manage to get 0.996 of pair completeness with more discriminative and smaller blocks. The $predicate2$ achieved 3,018 true matches and 0,654 of reductions ratio. The best metrics were achieved by our indexing techinique based on phonetic encoded term frequency. Despite the major number of blocks, this approach obtained 3,020 true matches doing less comparisons. These results reflected on reduction ratio and pair completeness, getting 0.992 and 0.996, respectively.

Tabela 3. Results of each indexing techinique used.

	predicate 1	predicate 2	term frequency
true matches retrieved	2,382	3,018	3,020
number of blocks	5,806	6,432	6,458
number of comparisons	44,406,049	29,111,755	645,800
reduction ratio	0.472	0,654	0,992
pair completness	0.786	0.996	0.996

Our proposal of phonetic encoded term frequency to establish the best candidade pairs for linkage comparisons revealed efficient in terms of accuracy and runtime execution. Another major contribution of this work refers to perform indexing while allow the comparison methods to encode or encript the blocks, considering the privacy-preserving concern of AtyImo tool.

5. Related Work

Record linkage tools frequently offer a set of methods to reduce the amount of pairwise comparisons potentially appearing in big data scenarios. Most popular tools [Schnell et al. 2004, Elfeky et al. 2002, Christen 2008] can provide a indexing based on a single attribute (traditional indexing), some sorted neighborhood strategies, canopy clustering and string-map based approaches. To each of these indexing methods, several parameters are available to be tested on user data in order to choose the best approach.

Several indexing techniques usually employed to decrease record linkage complexity were described and compared on [Christen 2012]. They applied different methods to link some real-world and synthetic datasets. Measures like reduction ratio, pair completeness, pair quality, and accuracy were utilized to assess the results. Their experiments showed that the number of parameters to be configured and the quality of the data to be linked make difficult a successful application of any indexing technique. Some of these techniques are also explained and evaluated in [Yeddula and Lakshmaiah 2016].

Traditional blocking techniques, as well locality-based ones are discussed and compared in [Steorts et al. 2014]. The authors have used synthetic data sets and evaluated different metrics (recall, reduction ratio, and complexity). They also discussed some privacy-preserving requisites related to blocking and indexing techniques.

Python provides the *recordlinkage* library that implements a full index approach based on the MultiIndex object provided in the Pandas library. This approch returns all pairwise combinations (product of the records present in both data sets). It is possible to provide a blocking key (column name) in order to reduce the number of blocks generated. The library also implements a sorted neighborhood approach.

In order to meet the requisites imposed by big data scenarios, AtyImo runs over the pySpark library and offer two predicate-based indexing approaches to split the data into blocks. These predicates put in the same block those records which agree with some rules, like first name and birth date, or last name and mother's name. Experiments show a good accuracy level using this approaches [Pita et al. 2018].

6. Conclusion and future work

Record linkage is a technique widely used in data mining and data warehousing applications to allow for the aggregation of data coming from disparate data sources. Big data scenarios impose significant challenges for data linkage tools. The growth of data sources and the need for increasing levels of accuracy open room to the development of novel solutions and the improvement of classical tools.

This work presented an extension of a cluster-based record linkage tool for indexing data using phonetic enconded term frequency. Results for accuracy and pairwise comparison are promising and outperforms existing, concurrent solutions. As future work, we plan to develop other techniques such as TF-IDF and unsupervised machine learning approaches.

Referências

Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE.

Binstock, A. and Rex, J. (1995). Metaphone: a modern soundex. *Practical Algorithms for Programmers. Addison Wesley*.

Christen, P. (2008). Febrl: a freely available record linkage system with a graphical user interface. In *Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80*, pages 17–25. Australian Computer Society, Inc.

Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, 24(9):1537–1555.

Christen, P. and Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining*, pages 127–151. Springer.

Elfeky, M. G., Verykios, V. S., and Elmagarmid, A. K. (2002). Tailor: A record linkage toolbox. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 17–28. IEEE.

Newcombe, H. B., Kennedy, J. M., Axford, S., and James, A. P. (1959). Automatic linkage of vital records. *Science*, pages 954–959.

Pita, R., Pinto, C., Sena, S., Fiaccone, R., Amorim, L., Reis, S., Barreto, M., Denaxas, S., and Barreto, M. E. (2018). On the accuracy and scalability of probabilistic data linkage over the Brazilian 114 million cohort. *IEEE Journal of Biomedical and Health Informatics*, 22(2):346–353.

Schnell, R., Bachteler, T., and Bender, S. (2004). A toolbox for record linkage. *Austrian Journal of Statistics*, 33(1-2):125–133.

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014). A comparison of blocking methods for record linkage. In Domingo-Ferrer, J., editor, *Privacy in Statistical Databases*, pages 253–268, Cham. Springer International Publishing.

Yeddula, S. and Lakshmaiah, K. (2016). Investigation of techniques for efficient and accurate indexing for scalable record linkage and deduplication. *International Journal of Computer & Communication Technology*, pages 24–30.