On the Use of Semantic Blocking Techniques for Data Cleansing and Integration

Jordi Nin[1], Victor Muntés-Mulero[2], Norbert Martínez-Bazan[2], Josep-L. Larriba-Pey[2]

[1]IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB s/n 08193
Bellaterra, Catalonia, Spain
Email: jnin@iiia.csic.es

[2]DAMA-UPC
Computer Arch. Dept. Campus Nord. UPC
C/Jordi Girona, 1-3. Módul D6. 08034 Barcelona
Email: {vmuntes, nmartine, larri}@ac.upc.edu
http://www.dama.upc.edu

***Abstract*— Record Linkage (RL) is an important component of data cleansing and integration. For years, many efforts have focused on improving the performance of the RL process, either by reducing the number of record comparisons or by reducing the number of attribute comparisons, which reduces the computational time, but very often decreases the quality of the results. However, the real bottleneck of RL is the post-process, where the results have to be reviewed by experts that decide which pairs or groups of records are real links and which are false hits.**

In this paper, we show that exploiting the relationships (*e.g.* foreign key) established between one or more data sources, makes it possible to find a new sort of semantic blocking method that improves the number of hits and reduces the amount of review effort.

Keywords: Semantic information, blocking algorithms, record linkage, data integration, data cleansing.

I. Introduction

The amount of information stored about individuals has increased dramatically in the recent years [17]. The ubiquitous presence of computers causes this information to be distributed and represented in a large amount of heterogeneous ways. Resolving the different instances of one entity among different heterogeneous data sources is, thus, a requirement in many cases.

As a consequence, the importance of tools and techniques that contribute to the process of data cleansing and data integration [20] has increased in the recent years. Among these, *Record Linkage* (RL) has gained relevance. The purpose of RL is to either identify and link different record instances of one entity that are distributed across several data sources, or to identify records from a single data source with similar information.

In practice, since the size of the source files is usually very large, comparing all the records among them becomes unfeasible. Therefore, RL resorts to blocking methods that are meant to gather all the records that present a potential resemblance, only allowing comparisons between records within each block. Typically, the traditional blocking methods for RL, like standard blocking [9] or sorted neighborhood [8] are based on the syntactic information of each record.

However, the quality of the results provided by these methods is very dependant on the data chosen to classify records in blocks and the quality of this data. A solution to this problem is to relax the creation of a block, by building larger blocks that allow for the comparison of a larger number of records. However, this has three clear drawbacks: (i) the number of unnecessary comparisons increases, (ii) the probability of RL to relate records belonging to different entities also increases and (iii), the review effort of the results grows and becomes a problem.

In this paper, we propose a new blocking method that builds groups of records based on the relationship among them in the data sources, as opposed to the use of the syntactic information of its attributes, as in other classic methods. In order to find the relationship of a record with other records, we build a collaborative graph [12] that contains all the entities that are related to the record under analysis up to a certain degree, using, for instance, foreign key relationships. The records included in the graph are the building units of each block and we link them using regular record comparison strategies. With the results presented in this paper, we show that, for environments where the connectivity between entities is low, our blocking method can clearly improve the quality of the results by significantly reducing the number of false hits while maintaining and, in the best cases, improving the number of returned real hits.

The structure of the paper is as follows. In Section II we introduce the basics of RL and standard blocking methods. Then, in Section III we present our approach based on semantic blocking. Section IV describes the experiments. The paper finishes with the description of some preliminary work, some conclusions and a description of future work.

II. Record Linkage: preliminaries

Record Linkage processes a set of files obtaining a list of record groups that can be considered similar. For our work, we consider that the RL process is formed by different phases, as shown in Figure 1. To start the process, data sources are cleaned or pre-processed in such a way that the attributes in the record files are normalized individually to allow a simpler comparison with other data in the following steps [3].

11th International Database Engineering and Applications Symposium (IDEAS 2007)
0-7695-2947-X/07 $25.00 © 2007 IEEE

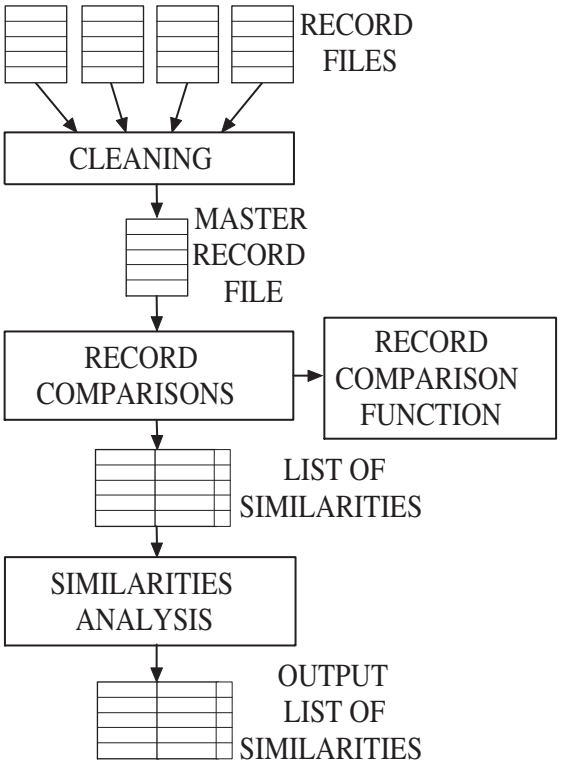

Fig. 1. Record Linkage processing model.

Once the pre-processing is done, we proceed with the RL analysis. There are two kinds of RL algorithms for record comparison. The first type is based on probabilistic methods and the second type is based on distance functions [19], [20]. During the record linkage process, the records are compared following a strategy that may have several objectives, like reducing the number of comparisons as with the Standard Blocking or Sorted Neighborhood methods, or finding the largest groups of similar records with the lowest comparison cost as in RAR [18]. Usually, the proposed blocking techniques build the blocks using the information obtained from the values in one or more attributes (syntactic information).

In order to avoid possible errors induced by the blocking methods, it is usual to perform several passes using different criteria. In any of those cases, a record comparison function is used, which returns a similarity weight W between pairs or groups of records.

After the RL process delivers the result, it is necessary to analyze the similarities. This last step usually requires the human intervention by means of expert individuals. If the number of pairs produced by RL is large, the time necessary to allow human experts to review all the pairs may be really significant leading also to extra unnecessary review errors.

In order to obtain a good result from the RL process, it is necessary to obtain the maximum number of real groups in the result, but it is also necessary to minimize the number of false positives, so that the manual process is also minimized.

The phase where records are compared is the most expensive in the RL computational process, with a quadratic complexity ($O(N^2)$) in the number of records N, as opposed to the linear complexity of the other phases. However, the analysis of similarities may be even more costly in time and budget because there may be different human reviewers involved in it during a considerable amount of time.

Blocking methods

In this subsection, we describe the most frequently used blocking methods: *Standard Blocking* (SB) and *Sorted Neighborhood* (SN). We explain the basics of these two classical methods and describe their main drawbacks.

1) Standard Blocking: The Standard Blocking method (SB) groups records that share the same *Blocking Key* (BK) [9]. An exhaustive comparison of all the records among them is performed within each group.

A BK is defined based on information extracted from one or more attributes. Usually, a BK can be either a common categorical attribute, *e.g. marital status* {single, married, divorced or widowed}, or a common numerical attribute, *e.g. age*. When files do not have common categorical or numerical attributes, a BK can be also a part of a string attribute, *e.g.* the first four characters of a *surname* attribute. The cost-benefit trade-off of the BK selection is studied in [19].

Note that using SB the size of the blocks is not constant. Therefore, in some situations the final blocks may contain a large number of records, in which case, the amount of work will not be reduced much from the comparison of all the records without using blocks, but the possibility to match really similar records will grow. However, the amount of false positives will most probably be large.

2) Sorted Neighborhood: The Sorted Neighborhood (SN) method [8] sorts the records based on a sorting key (SK) and then moves a so called Sliding Window (SW) of fixed size l sequentially over the sorted records. At every step, the oldest record is removed from the window and a new record is inserted, which is compared to all the records in the window.

An important problem with SN arises if a certain number of records, larger than the window size, have the same value in a SK. For instance, let us suppose that we are using SN with two similar files based on a SK extracted from an attribute *'surname'*. Typically, if the data sources are large enough, there will be thousands of records containing the value *'Williams'* or *'Smith'* in that attribute and, therefore, not all the records with the same value in a SK will be compared.

Another problem with SN occurs when there is a significant error rate where the differences among data may cause the sorting algorithm to separate the values enough to avoid their comparison.

III. Semantic Blocking for RL

We propose a new family of blocking algorithms that substitute the blocking or sorting key used by typical blocking methods by another type of block building method based on the *context*. This method is not based on the values of

0-7695-2947-X/07 $25.00 © 2007 IEEE

one or more attributes of the records but on the relationship established between the records in the source files. We refer to this relationship as *context or semantic information*. As stated by the *Context Attraction Principle* presented in [10], the basic idea is that two occurrences that could belong to the same individual are more likely to refer to a single individual if they are closely related, in the context established by their relations with other entities in the data set.

The method used to relate the entities of a database will vary depending on the data sources. For example, if we have a relational database as data source, we are able to obtain some context information from the foreign keys of the database, which is the case studied in this paper. If we have a plain text file as a data source, we can relate the entities based on the relationships between all the records that contain values with the same contextual meaning in one or more textual attributes. Or even, if we turn to graph databases [7] [11] [16], we can use the edges between entities to infer similarities between the text items in the nodes.

This approach is very useful in databases that contain a lot of information about the relationship of an individual with the rest of the database, specially when the connectivity between entities is not very high, as we will show below. There are several examples of databases that have this kind of information. Bibliographic databases like Citeseer [5] or social network databases that proliferate in the web nowadays are examples of data sources that contain a large quantity of information about the relationship between entities.

The Semantic Blocking that we propose allows to avoid the problems caused by spelling errors in the classic blocking approaches described above, by only allowing comparisons between records that have a relationship among them. This reduces the chances to gather in the same block records that are similar but do not have any relation.

A. Semantic Graph Blocking

The Semantic Graph Blocking (SGB) proposed in this paper is based on the capabilities that collaborative graphs offer in order to extract the information about the relationship between the records in the source files. Collaborative Graphs are a common method for representing the relationships among a set of entities [12]. Nodes represent the entities and edges capture the relationships between entities.

The main idea of our approach is to build up a graph, exploring the relationships of the entity that we want to deduplicate or merge. We denote our graph as $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each element in V represents an entity and each element in E represents a relationship established between two entities, e.g., the information inferred from the foreign keys in the database.

In order to build the graph, we create a first node $v_0 \in V$ that represents the current record to be compared to other records to find pairs corresponding to the same entity. Following we create a new node v_i for every record that is directly related to the first one. From this first level of nodes, we explode their relationships again, adding all those records that did not previously exist in the graph. This process is repeated until there are no more nodes to explode. However, since theoretically all the nodes could be connected and, therefore, all the records would be included in the same block, we reinforce the stop condition either by predefining the maximum depth of the graph or by specifying a maximum number of nodes.

Once the block is created, we use regular record and attribute comparison functions to detect similarities between records, comparing all the records in a block between them. Note that, with our technique, we only need to execute this process once. More passes would be unnecessary because SGB does not depend on possible misspelling in the data.

Figure 2 shows a small graph built from a toy database containing three relations in a star schema [4]. Relation *AUTHORS* is a table containing a set of authors. For each author, the relation contains an identifier, the name and other attributes that are not important for this example. Relation *PAPERS* is a table that contains a set of publications. Analogously to *AUTHORS*, each paper has an associated identifier. Finally, the table *WRITES* relates the authors to all those papers written by them. In this example, the purpose is to identify duplicated authors in the database. There are three possible authors that could be considered the same entity in a record linkage process based on the edit distance: *Smith, John*; *Smith, Jehn* and *Xmith, Jhon*. However, suppose that there are two real authors whose names are *Smith, John* and *Smith, Jean*.

In order to find the duplicated information, we start the record linkage process. For each author, a graph is created by following the relations established by the foreign key attributes in table *WRITES*. In Figure 2, we have limited the depth of such a graph to distance 2. Of course, the size of this graph is very small because of the size of this toy example. In a real system, a graph might contain hundreds of nodes. The four-nodes graph obtained by the first author in relation *AUTHORS*, *Smith, John*, is marked in a darker area. We can observe that the other three authors included in the graph are those with identifiers 4, 5 and 6 respectively. The remaining nodes are *too far* in this domain to be considered related to *Smith, John*. Once the block is created by obtaining the nodes included in the graph, we compare all the names in this block. In this case, the record linkage process would consider *Smith, John* and *Xmith, Jhon* to be the same entity, while *Smith, Jehn* would be discarded as a possible duplicated value of the same entity.

Note that, using a traditional record linkage process based on standard blocking or sorted neighborhood, we would not have any information that would help us to understand whether these three names refer to the same unique author. Thus, when they are sorted by name, *Smith, John* and *Smith, Jehn* would probably be considered to refer to the same author (and they are not). On the other hand, *Xmith, Jhon* would never be compared to the other two using SB or SN, since there is a mistake in the first character, which would most probably place the first two occurrences in a different block from that of this third occurrence.

0-7695-2947-X/07 $25.00

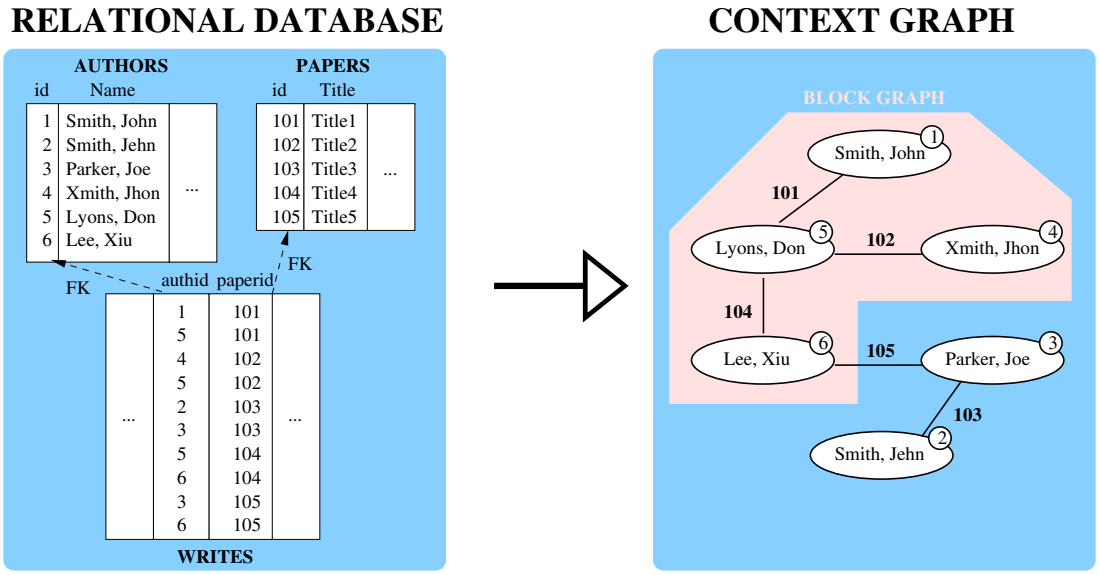

Fig. 2. Toy example of Semantic Blocking.

IV. Experiments

In order to test our approach in a wide variety of scenarios, we have divided our experiments into two parts. First, we test the quality of our approach compared to other traditional approaches using a set of synthetically created databases. This experiment allows us to test different conditions such as the size of the data set or the degree of connectivity between entities. Second, we test our entity resolution technique using the Citeseer database [5], presenting a real scenario to run our new RL blocking technique.

All the experiments have been performed on a 64 bit Intel Core2 Duo, at 2.6 GHz, using 2 GB of RAM.

A. Metrics

For each experiment performed in this paper, we analyze the quality of the results using the typical quality measures: recall, precision and F-measure [15]. Precision is defined as:

$$\text{precision} = \frac{|\{\text{real pairs}\} \cap \{\text{retrieved pairs}\}|}{|\{\text{retrieved pairs}\}|} \quad (1)$$

Recall is defined as:

$$\text{recall} = \frac{|\{\text{real pairs}\} \cap \{\text{retrieved pairs}\}|}{|\{\text{real pairs}\}|} \quad (2)$$

Sometimes it is desirable to have one single number for the performance of an algorithm instead of two. In such cases, the F-measure is frequently used [14]. It can be parameterized to give a higher weight to either precision or recall. The neutral parametrization, where precision and recall are weighted equally, is used throughout this work. Thus, F is defined as the weighted harmonic mean of precision and recall:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (3)$$

Name	Standard Blocking (SB)	Sorted Neighborhood (SN)	Semantic Graph Blocking (SGB)
Par1	substring(Full name,0,4)	l=200	S=200
Par2	substring(Full name,0,3)	l=500	S=500
Par3	substring(Full name,0,2)	l=1000	S=1000

TABLE I
Different parameterizations for blocking (SB), sorted neighborhood (SN) and Semantic Graph Blocking (SGB).

B. Parameterizations of the methods

We have tested the three blocking methods using three different parameterizations for each one. We show the parameterizations in Table I. We consider small, medium and large blocks. The second column in the table shows the three different BK selected for each parametrization. As we can observe, the BK in the first parametrization is expressed as $substring(Full\,name, 0, 4)$ meaning a substring containing four consecutive characters of attribute *Full name* starting at string position 0. The other two parameterizations correspond to those strings containing the first three and the first two characters, respectively. In the third column of the same table, we show parameter l that defines the window size in the SN blocking. The fourth column defines the values selected for S, that defines the maximum size in terms of nodes of the collaborative graph.

We use two different thresholds to accept the similarity of a pair of records using their edit distance [13]:

- **Strict RL:** we force RL to classify as a hit only those pairs of records that differ less than 25% in their full name, i.e., it is necessary to change less than 25% of the characters of one of the strings in the pair to let it be equal than the other string in the pair.
- **Weak RL:** we force RL to classify as a hit all the pairs

of records that differ less than 50% in their full name.

The reason for using different thresholds for RL is that we want to see the influence of such parameter on the quality of the result and, therefore, on the reviewing effort, for the different methods.

C. Synthetic Data Experiments

First, we will test our blocking technique through a variety of bibliographic data sets generated synthetically. The synthetic data sets contain information about authors and papers, and the relationship between them, following the schema used in the toy example of Figure 2. This will allow us to understand the effects of our technique when different properties in the data set are modified. We test the different blocking approaches for data sets of different sizes. Specifically, we have generated data sets containing 10000, 100000 and 250000 authors. For each of these sizes, we try different author duplicate percentages, namely 50%, 20% and 5% of authors are duplicated in each case. The number of erroneous characters for each case is calculated using a normal distribution $N(2,1)$ with mean $\mu = 2$ and standard deviation $\sigma = 1$. The number of duplicates per each original duplicated entity is distributed following a $N(3,1)$ distribution. We control the degree of connectivity by changing the average number of papers per author. Specifically, we build data sets were the number of papers per author is distributed as $N(2,1)$ for a first set of experiments and $N(20,1)$ for a second set of experiments. The number of authors per paper is normally distributed following a distribution $N(3,1)$. The distance between an original entity and its duplicates follows a normal distribution $N(3,1)$.

In order to create a realistic scenario, we have used names and surnames extracted from a frequency dictionary containing 1564 names and 13068 surnames.

Following, we analyze the effect of varying the different factors studied on the three measures used to evaluate the blocking methods, namely recall, precision and F-measure.

1) Effect of the data set size: Figure 3 shows the effect of the variations of the data set size on the different quality measures. The figure is divided into two rows of plots. The upper row shows the results when applying the strict RL, while the lower plot shows the results for the weak RL. The leftmost plot in the upper row shows the effect of the size of the data set on the recall for the three methods and the three parameterizations for each method. First of all, we can observe that the average recall of SGB is higher than that using other blocking methods, independently of the size. In addition, as the size of the data set increases, SN tends to decrease in terms of recall. This is because, the size of the window is fixed and the number of elements with the same value for a certain attribute increases. These two effects reduce the chances to find different instances of the same entity in a certain window. For SB, increasing the size of the data set does not have a radical effect since all the instances within a block are compared among them. Note that although the recall is not affected, the size of the blocks will increase significantly. On the other hand, using SGB, even keeping the size of the block, the recall is kept similar. Note that the fact that new authors (different entities that can have small edit distances with existing authors in the data set) are introduced into the database will not prevent the SGB blocking algorithms to find the closest relations in the graph. Also note that we are not increasing the connectivity of the existing entities, but just adding new entities and preserving the average degree of connectivity. The effect of altering the connectivity is studied later in this section. The leftmost plot in the lower row shows the same results for weak RL. As expected, we can observe that the recall slightly increases, since we accept a larger number of pairs. Of course, this will have a negative effect on the precision, as we will see later on.

If we focus on the two plots in the middle, we observe the effect of the size on the precision. We can clearly see that, while precision using SB and SN is clearly reduced when the size of the data set increases, it is kept constant for SGB. Again, having a larger number of authors but keeping the average degree of connectivity between them does not prevent SGB from finding the same relationships as with a smaller data set. Therefore, precision and recall are preserved. For weak RL the differences are even more noticeable. Strings in SB or SN are compared because they are relatively close in terms of the blocking key or the sliding window, while for SGB they are compared because they are related somehow in the data, even though they are completely different. Therefore, relaxing the record linkage acceptance condition will have a big impact for SB and SN, accepting a clearly larger set of pairs, and naturally reducing precision. On the other hand, it will not affect so clearly SGB since the strings to be compared in this case are in general clearly different and are not accepted as pairs even using weak RL.

In general, the differences between the three methods are summarized by the F-measure, shown in the rightmost plots in the figure. Again we can see that an increase in the size of the data set does not affect significantly our technique, while it degrades the quality of the results obtained by SB and SN. As we have seen previously, this effect is more marked when we relax the restrictions to accept a pair as valid.

2) Effect of the percentage of duplicated entities in the data set: Next, we study the effect that the portion of entities with duplicated entries has on recall, precision and the F-measure for the three studied techniques. Our experiments show that the portion of entities having duplicates does not have a significant effect on the quality of the results obtained. Figure 4 shows the average F-measures for different percentages of duplicated entities for strict RL. As we can observe, there is not a clear trend in the quality indicator. Other experiments not plotted in this figure show that we can extract the same conclusions by observing recall and precision separately for both the strict and the weak RL.

3) Effect of the connectivity of the entities in the data set: Finally, in Figure 5 we study the effect of the average connectivity of an author on recall, precision and the F-measure. As we explained in the experiments setup, we study two different situations. In the first case, we assume that the

0-7695-2947-X/07 $25.00

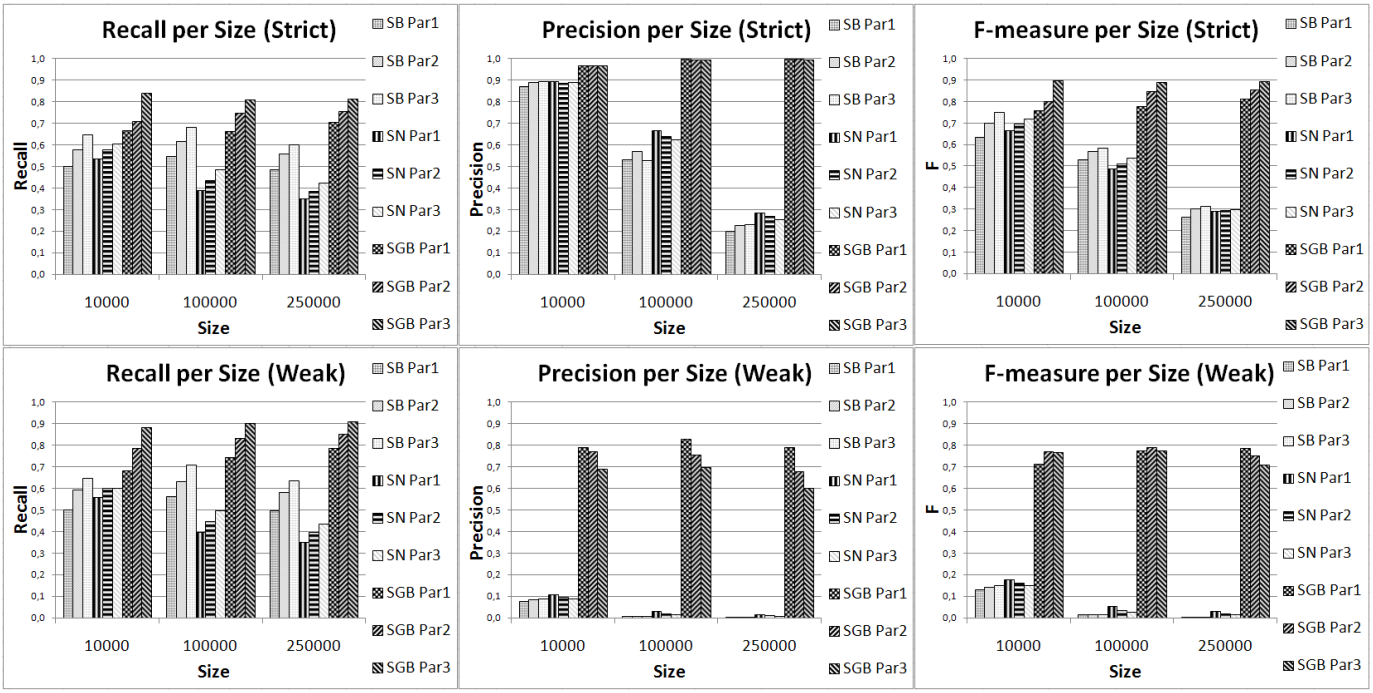

Fig. 3. Effect of the data set size on the recall, precision and F-measure.

distribution of the number of papers per author is centered in 2, while in the second case it is centered in 20. Note that the average number of papers per author in a real database like Citeseer is 2.76, therefore, the first scenario is an example closer to the reality and the second scenario aims at simulating an unrealistic highly connected scenario to extend our study.

In the two leftmost plots in Figure 5, we can observe the effect of the connectivity between entities in the graph on the recall. Both for strict RL and weak RL, we can see that the recall is clearly better when the connectivity between authors is not very high, i.e., when the number of papers per author is reduced. On the other hand, when we increase the connectivity, in this case by increasing the average number of papers per author, the recall for SGB is reduced and

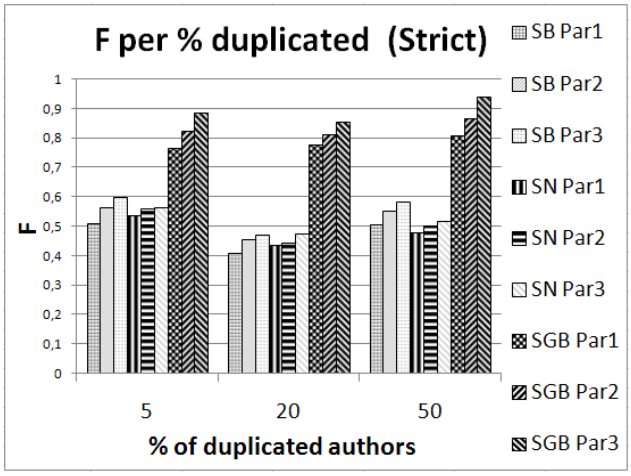

Fig. 4. Effect of the portion of duplicated entities on F for Strict RL.

cannot be considered significantly better than that for SB or SN. Since in a bibliographic database two duplicates of an entity would never be connected directly, because they cannot coauthor a paper together, if the connectivity is high, a large part of the blocking graph, which has a limited size, will consist of all those entities directly connected to the explored entity, therefore reducing the probabilities of SGB finding real matches.

However, the precision is not affected as clearly as the recall. The two plots in the middle show us that SGB achieve better precision than the other two methods both when the connectivity is low and high. The F-measure in the rightmost plots, show that combining recall and precision, SGB would still be the blocking method of choice.

Note that, in our work, we are focused on a bibliographic database-like scenario. In other words, we are assuming an scenario were the entities in the graph are not highly connected. In other situations where the connectivity between the entities in the graph would be higher, SBG would not be suitable. We have tested an unrealistic scenario where we had 10000 authors and each paper had an average of 1000 authors to simulate a highly connected environment. In this situation, the number of authors directly connected to the entity we are trying to deduplicate will be, in general, larger than the number of nodes included in the block graph and, therefore, SGB will be unable to find most of the real pairs, having a very low recall. Therefore, the technique we are presenting must be used in those scenarios were the connectivity is low.

D. Citeseer Experiments

Now, in order to test our approach in a real scenario, we have performed some experiments on the Citeseer database [5].

0-7695-2947-X/07 $25.00

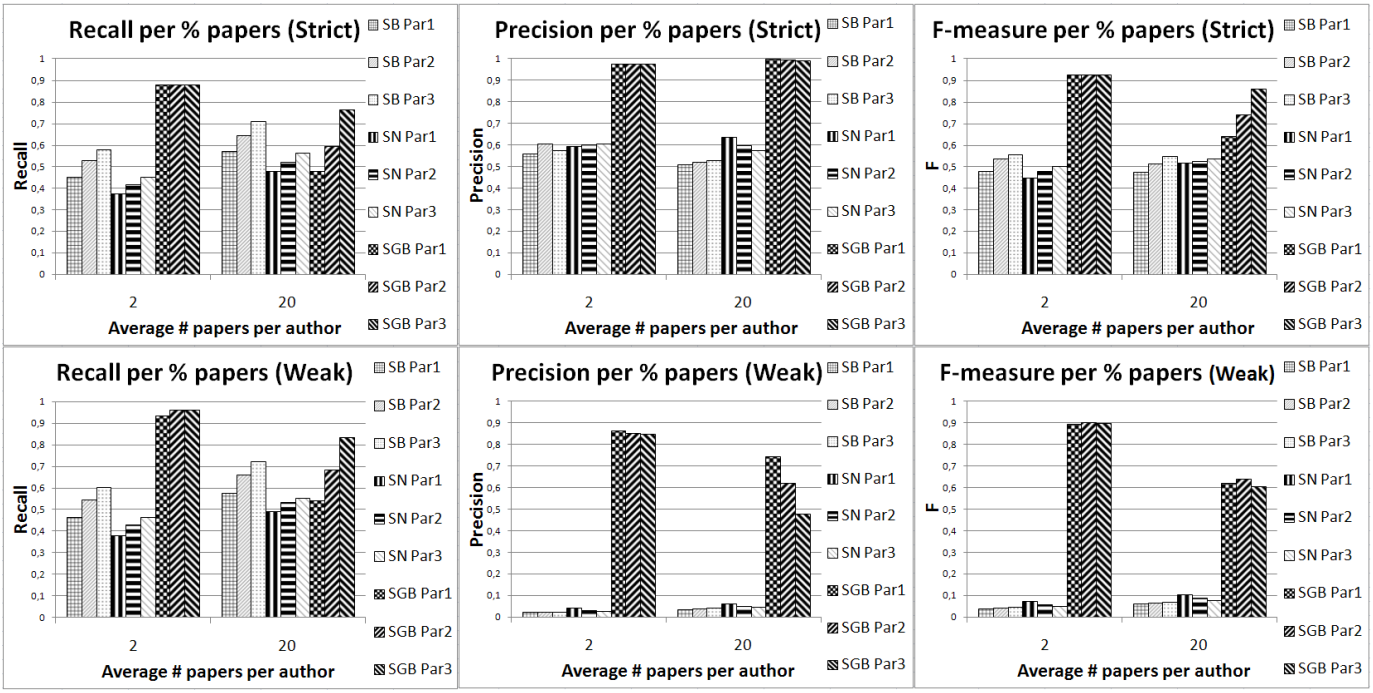

Fig. 5. Effect of the average number of duplicates author with duplicates on the recall, precision and F-measure.

Citeseer is one of the most used scientific databases with more than 1,6 Million scientific papers and more than 750,000 authors. We have only downloaded a part of all the attributes stored in Citeseer: the full name of the authors and the identifier of the papers. Table II shows a sample of the downloaded file. As we can see, our approach only needs two attributes to exploit the semantic information stored in the Citeseer database. We are able to construct a collaborative graph using the author full name as the node and the paper identifier as the relationship between two individuals (or records).

The Citeseer database presents a scenario where data are automatically extracted from a set of heterogeneous data sources presented in different formats. In this kind of scenarios, it is very difficult to preserve data consistency and, therefore, the probability of having mistakes in the integrated data set is very high. We can observe this in Table III, where we present all the occurrences of the ten full professors of the department of Computer Architecture at UPC, and the different names introduced in the database for each of them.

In order to compare the quality of the methods that we are working with, we focus exclusively on the names appearing in Table III. For each name we create a block and analyze the quality of the results obtained for that block, namely the recall and the precision.

We observed that there are several occurrences of the same individual differing in more than 25%, *e.g.* J.M. Llaberia is the same author than Mara Llabera[1]. This allows to study two scenarios by using weak and strict RL.

Results: We have used recall and precision as the metrics to assess the methods evaluated. In addition, we take into consideration the absolute values of the number of pairs provided by the different methods in order to assess the real amounts both in terms of quality and in terms of review work needed by human experts.

Table IV shows the accumulated results for Citeseer of the experiments run for the full professors of the Computer Architecture Department at UPC shown in Table III. The real total number of duplicates for our data set is 38 and the recall (R) in Table IV is calculated based on this figure. Both the recall and the precision (P) are calculated as percentages and the strict and weak RL are separated into two groups of columns in the table.

We can observe two different trends in the figures of Table IV. First, weak RL improves the recall of SGB over strict RL. The reason is clear, the number of records with a small edit distance per block created with SGB is relatively reduced, and with high chances, if they are similar syntactically, they will most probably be the same, confirming what we observed in the synthetic experiments. Thus, relaxing the RL from strict

Full name	Paper identifier
Gabriele Scheler	1
Xianshu Kong	2
Hazel Everett	2
Godfried Toussaint	2
Helge Frauenkron	3
Peter Grassberger	3

TABLE II
SAMPLE OF THE FILE DOWNLOADED FROM CITESEER [5].

[1]The real name is *José María Llabería*. In the first case *José María* has been abbreviated to *J.M.* In the second case *José* has been removed possibly due to a parsing error and the fact that the vowels contain an stress sign (í), that has been omitted, generating a significantly different name (*Mara Llabera*)

Full Professor	Duplicated Records
Eduard Ayguade	E. Ayguade, E. Ayguad, Eduard Ayguad, Eduard Ayguad E Eduard Ayguad Parra*, Eduard Ayguade Parra*, Eduardo A. Parra
Jordi Domingo-Pascual	J. Domingo-pascual*, Jordi Domingo Pascual
Jordi Garcia	J. Garcia, Jorge Garca, J. Garcia-vidal*, Jorge Garcia-vidal*
Antonio Gonzalez	Antonio Gonz Alez*, Antonio Gonzlez, A. Gonzlez*, Antonio Gonzz*
Jesus Labarta	J. Labarta, Jess Labarta, Jes Us Labarta, Jesffs Labarta*, Jes Labarta*
J. M. Llaberia	Jos Mara Llabera, Jos M. Llabera, Mara Llabera*, J. M. Llabera
Manel Medina	Manuel Medina
Juan J. Navarro	J. J. Navarro, Juanjo Navarro*, Juan J. Navarroy*
Mateo Valero	Andmateo Valero, M. Valero, Larriba-pey Mateo Valero, Advisor Mateo Valero*, Mateo Valeroy
Miguel Valero Garcia	Miguel Valero-garc, Miguel Valero-garca, M. Valero-garca*

TABLE III

DUPLICATED RECORDS FOR FULL PROFESSORS OF THE COMPUTER ARCHITECTURE DEPT. OF UNIVERSITAT POLITÈCNICA DE CATALUNYA. *THESE AUTHORS ONLY APPEAR IN ONE DOCUMENT.

Method		Strict RL				Weak RL			
Type	Parame–trization	Pairs provided	True matches	R	P	Pairs provided	True matches	R	P
SB	Par1	18	12	31.6%	66.7 %	435	16	42.1%	3.7%
SB	Par2	22	17	44.7%	77.3 %	808	17	44.7%	2.1%
SB	Par3	22	17	44.7%	77.3 %	1669	21	55.3%	1.3%
SN	Par1	14	11	28.9%	78.6%	241	15	39.5%	6.2%
SN	Par2	15	12	31.6%	80.0%	393	16	42.1%	4.1%
SN	Par3	21	16	42.1%	76.2%	480	20	52.6%	4.2%
SGB	Par1	16	14	31.6%	87.5%	24	20	52.6%	83.3%
SGB	Par2	16	14	31.6%	87.5%	27	21	55.3%	77.8%
SGB	Par3	16	14	31.6%	87.5%	32	22	57.9%	68.8%

TABLE IV

ACCUMULATED RESULTS OF THE TWO METRICS FOR THE EXPERIMENTS PERFORMED.

to weak, we have more chances to catch similar records with a lower probability of finding false positives. Specifically, the number of false positives is very small, less than 33% with strict RL, which amounts to only 10 elements, a very insignificant effort for an expert human reviewer.

On the contrary, for SB and SN, the weak RL improves the recall, while it reduces the precision over strict RL significantly. Note in this case that the precision is reduced because the number of similar records found by the methods grows significantly to hundreds of pairs, at least, while the number of real similarities is still kept low. This larger number of false positives implies a large number of cases for the expert human reviewer, leading to a larger effort and number of possible human errors compared to strict RL.

If we compare the methods, we find that SGB is able to capture a larger number of true matches when relaxing the RL than SB and SN. Also, it can be compared in terms of recall to the other methods even for strict RL situations. This makes us think that semantic blocking (i.e. SGB) is better than SB and SN if there are such relationships present in the data set. Also, the review effort imposed by SGB is significantly smaller (two orders of magnitude in some cases) than the review effort imposed by SB and SN. This makes us state that SGB is significantly better than SB and SN because of the smaller chances to accept false positives, both in an automated review effort and in a manned review process. As an example of the benefits obtained by SGB, if an expert needs 10 seconds on average to decide whether a hit is real or false and we are interested in obtaining as many real hits as possible, (i.e. we choose Par3 for the three methods using a Weak RL), the whole RL plus reviewing process would take around 4.6 hours using SB, 1.3 hours using SN and 5.33 minutes using hour approach. In addition, with our approach we would obtain a slightly larger amount of real hits than with the other two approaches. Note that the record linkage process using SGB takes less than a second to be executed.

Finally, the computational effort of using SGB might be similar to that imposed by SB or SN. The former requires the traversing of relationships by means of foreign keys in a database or a set of tables. The latter impose the sorting of the data set to create the blocks. Again, note that while we must typically perform several passes with SB or SN, only one pass is necessary when using SGB. Of course, this will finally depend on the size of the blocks created by each blocking technique. A detailed performance analysis and the implications on the precision and recall are out of the scope of this paper.

V. RELATED WORK

Among the large amount of work done in the area of Record Linkage, there is also a considerable effort in the use of graphs and relations to relate entities and disambiguate references. Apart from the references cited in the previous chapters which give a taste of the work done in the area of standard blocking for Record Linkage, in this section we give a short description of a few pieces of research that are related to the use of graphs or relationships for the disambiguation task.

0-7695-2947-X/07 $25.00 © 2007 IEEE

Among the many different pieces of work in the area, we can distinguish Kalashnikov and Mehrotra's [10], who analyze the inter-object relationships to improve the quality of reference disambiguation. In some other work by Ananthakrishna et al. [1], the authors use the similarity of directly related links to solve record deduplication that can be applied to hierarchical relationships. Bhattacharya and Getoor [2] use an object consolidation method that goes beyond that of [1] allowing other types of relationships. Our work differentiates from the other works that take into account relationships for data disambiguation or record linkage in that we use the inter-object relationships, no matter how distant they are, to build blocks. The records in those blocks are compared among them with record comparison functions as in a standard blocking strategy.

VI. Conclusions and future work

In this paper, we have presented a new sort of blocking methods based on the context information extracted from the relation among entities in the database. We present an specific method called Semantic Graph Blocking (SGB) oriented to reduce the expert review time in the RL process, using a collaborative graph to build blocks based on contextual information. We show the results of our SGB in comparison to the classic blocking and sorted neighborhood blocking methods using a synthetically created set of databases and the Citeseer reference database.

We have shown that our approach tackles the most important problem in data integration and cleansing: the time consumed in the post-process of RL and, at the same time, it improves the amount of hits (recall) compared to the classic blocking methods, when the connectivity between entities in the data set is low. We show in our experiments that we can drastically reduce the false hit ratio and, in the best cases, we reduce the number of false positives by two orders of magnitude. In addition we improve the quality of the RL process when we relax the hit acceptance threshold.

As future work we want to explore different approaches for the SGB, using strategies that allow to prune the graphs obtained without reducing the hit ratios. In particular, we want to explore the differences between creating blocks based on levels or number of nodes included. Also, we want to explore the possibility of using comparison weights based on the distance between the nodes compared. Finally, we want to explore the combination of syntactic and semantic blocks in order to improve the recall and the precision measures.

Acknowledgments

The authors want to thank Generalitat de Catalunya for its support through grant number GRE-00352 and Ministerio de Educación y Ciencia of Spain for its support through grant TIN2006-15536-C02-02. Jordi Nin wants to thank the Spanish Council for Scientific Research (CSIC) for his I3P grant.

References

[1] Ananthakrishna, R., Chaudhuri, S., and Ganti, V. Eliminating Fuzzy Duplicates in Data Warehouses. In Proc. of the VLDB Conference, 2002.

[2] Bhattacharya, I. and Getoor, L. Iterative record Linkage for Cleaning and integration. In Proc. of the DMKD Workshop, 2004.

[3] Bilenko, M., Basu, S., Sahami, M., Adaptive Product Normalization: Using Online Learning for Record Linkage in Comparison Sopping. Proceedings of the 5th Int'l. Conference on Data Mining 2005. Pages 58–65.

[4] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. SIGMOD Record, 26(1):6574, 1997.

[5] Citeseer database, http://citeseer.ist.psu.edu

[6] Goffman, Casper (1969). "And what is your Erdos number?". American Mathematical Monthly 76

[7] M. Gyssens, J. Paredaens, van den Bussche, J. and D. van Gucht. A graph-oriented object database model. IEEE Transactions on Knowledge and Data Engineering, 6(4). Pages 572-586. 1994

[8] Hernandez, M., Stolfo, S., (1998), Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 1(2), 1998.

[9] Jaro, M. A., (1989), Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Society, 84(406):414-420, 1989.

[10] Kalashnikov, D. and Mehrotra, S.. Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph, ACM TODS, Vol. 31, No. 2, June 2006, pp. 716-767.

[11] N. Kiesel, A. Schuerr and B. Westfechtel. GRAS, a graph-oriented (software) engineering database system. Information Systems, 20(1). Pages 21-51. 1995.

[12] Kubica, J., Moore, A., Cohn, D., Schneider, J., (2003) Finding Underlying Connections: A Fast Graph-Based Method for Link Analysis and Collaboration Queries. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

[13] Levenshtein, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady. Pages 707-710. 1966.

[14] C. J. van Rijsbergen, Information Retrieval, London, Butterworth, 1979, pp. 174 ff.

[15] G. Salton and M. McGill. Introduction to Modern Information Retrieval. McGraw- Hill, New York, NY, 1983.

[16] Srinath Srinivasa, Martin Maier, Mandar R. Mutalikdesai, Gowrishankar K. A. and Gopinath P. S. LWI and Safari: A New Index Structure and Query Model for Graph Databases. COMAD, pages 138-147. 2005.

[17] Sweeney, L., (2001), Information explosion, in Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies, eds. P. Doyle, J. I. Lane, J. M. Theeuwes and L. M. Zayatz, Elsevier, 43–74.

[18] Sung, S. Y., Li, Z., and Peng, S.. A Fast Filtering Scheme for Large Database Cleansing. International Conference on Information and Knowledge Management (CIKM), McLean, Virginia, USA, 2002.

[19] Torra, V., Domingo-Ferrer, J., (2003), Record linkage methods for multidatabase data mining, Information Fusion in Data Mining, Springer, 101-132.

[20] Winkler, W., (2003), Data Cleaning Methods, Proc. SIGKDD 2003, Washington.