Knowl Inf Syst (2011) 27:1–21
DOI 10.1007/s10115-010-0285-0

SURVEY PAPER

Approximate data instance matching: a survey

Carina Friedrich Dorneles · Rodrigo Gonçalves · Ronaldo dos Santos Mello

Received: 18 December 2008 / Revised: 20 October 2009 / Accepted: 16 January 2010 /
Published online: 9 April 2010

Abstract Approximate data matching is a central problem in several data management processes, such as data integration, data cleaning, approximate queries, similarity search and so on. An approximate matching process aims at defining whether two data represent the same real-world object. For atomic values (strings, dates, etc), similarity functions have been defined for several value domains (person names, addresses, and so on). For matching aggregated values, such as relational tuples and XML trees, approaches alternate from the definition of simple functions that combine values of similarity of record attributes to sophisticated techniques based on machine learning, for example. For complex data comparison, including structured and semistructured documents, existing approaches use both structure and data for the comparison, by either considering or not considering data semantics. This survey presents terminology and concepts that base approximated data matching, as well as discusses related work on the use of similarity functions in such a subject.

Keywords Instance data matching · Similarity function · Similarity matching · Record linkage · Record matching · Duplicate detection · Object matching · Entity resolution

This work was partially done while the author C. F. Dorneles was a Ph.D. student at the Informatic Institute, UFRGS, and was supported by Capes.

C. F. Dorneles (✉) · R. Gonçalves · R. dos Santos Mello
Centro Tecnologico (CTC), Depto. Informatica e Estatistica (INE),
Universidade Federal de Santa Catarina (UFSC),
Campus Universitario Trindade, Florianopolis, SC, Brazil
e-mail: dorneles@inf.ufsc.br

R. Gonçalves
e-mail: rodrigog@inf.ufsc.br

R. dos Santos Mello
e-mail: ronaldo@inf.ufsc.br

1 Introduction

Data matching is the process of bringing together data from different, and sometimes heterogeneous, data sources and comparing them in order to find out whether they represent the same real-world object. Since data come from different sources, it is expected they differ from each other in the way they are represented. This is a complex problem, since it is not trivial to assert that two heterogeneous data instances represent the same object of the reality. Heterogeneity can happen in data structure as well as in data value. Therefore, a data matching process must be able to analyze both structure and data value.

Data matching processes can be applied to several data management research areas, such as query/search over data sources (databases or Web data sources), integration of heterogeneous data sources, and so on. More specifically, query/search area have sub-areas such as similarity-based queries processing (or approximate queries) and query languages that are designed to query uncertainty. Data integration area has sub-areas such as data cleaning, duplicate data reduction, and so on. In approximate queries, the problem is to identify data instances which represent the same real-world object that a user specifies in the query. In data integration, the problem is to recognize different representations of the same real-world object based on data originated from different sources. In this context, data cleaning, for example, relates to the problem of detecting and correcting errors in data received from different data sources. In both cases, query and integration, the data may contain inconsistencies like different conventions, missing fields, typographic, typing and grammatical errors, which must be solved before being stored. Such situations are very common in modern databases, especially in cases where data are generated by various people or softwares.

In this survey, we qualify "approximate data matching" in two basic groups: (i) those which compare data based on data values; and (ii) those which compare data based on their structure, exploiting and extracting relevant data to the comparison. We review both categories, identifying many different approaches followed by the related work, as well as presenting a comparative analysis. It is important to note that Elmagarmid et al. [42] present a good survey on data matching, but they emphasize approaches that perform data value analysis rather than structure analysis. Furthermore, in this survey we focus only on works that rely on a similarity function (or distance function) when executing the data matching process. Other works in the literature use the probabilistic theory or statistical methods [86], for example, in order to manage heterogeneous data [47,93,98]. Furthermore, the works we present in this survey deal with structured data (relational tuples, records, XML, and so on). For proposals that consider unstructured documents, and use some specific algorithm in order to match the documents, see [5,92].

This paper is organized as follows: In Sect. 2, some basic concepts, which are important to understand some techniques, are described and a taxonomy for approaches related to approximate data matching is presented. This classification does not intend to define a strict taxonomy for the subject, but, instead, to organize and to structure existing related work. Sections 3, 4 and 5 introduce approximate data matching processes, and their related work, based on our taxonomy. Specifically, Sect. 3 presents proposals that use basically the data content in order to match instances. Section 4 describes proposals concerning comparison based purely on how the data is structured, and Sect. 5 concerns those work that consider both structure and content during data matching process. Section 6 describes some applications to approximate data matching. An overall comparison is provided in Sect. 7. In Sect. 8, a conclusion and suggestion of future work on the subject are presented.

2 Basic concepts and taxonomy

This section presents some basic concepts and a taxonomy of approximate data matching approaches that guide the rest of the paper.

2.1 Similarity measure

A data matching process employs a similarity function (or even a distance function) to compare data instances, given that there is not an exact way to assure that two heterogeneous data instances represent the same real-world object. A similarity function $fs(v1,v2) \rightarrow s$ establishes a score s for a pair of values $v1$ and $v2$, where s is in the interval [0,1]. The higher the score value, the more similar $v1$ and $v2$ are. Alternatively, a distance function $fd(v1,v2) \rightarrow s$ calculates a score s to a pair of values $v1$ and $v2$, where s is in the interval [0,infinity]. The closer the value is to zero, the more similar $v1$ and $v2$ are. The advantage of using similarity functions is to deal with a finite interval for the score values.

Two objects are considered to be similar, i.e., are considered to represent the same real-world object, if the similarity score surpasses a predefined *threshold* value. However, the choice of a threshold value is a difficult matter. This problem is not treated in this survey, being the focus of some related works [14,85,104].

2.2 Taxonomy for approximate data matching

Digital data may have several representations from a raw and unstructured information, available, for example, in a *Web* data source, without any metadata to provide a meaning for that, to a structured and strongly typed relational database. On considering or not their nature as well as an optional associated model, any existing data have a *content*, like a tuple in a relational database or a record file. A data content, or a data occurrence about a real-world object, is usually called a *value*. Besides the content, data usually have an *structure* that rules its organization. This structure provides the necessary basis to store and access the data instance, and sometimes allows a better understanding of its meaning.

We introduce such a foundation about digital data to propose a comprehensive taxonomy for approximate data matching approaches. In fact, we classify related work that performs data comparison for approximate matching purposes into three main categories, as described in the following:

1. Content-based matching: approaches that focus on data values, using a similarity function to compare two data instances. The value of a database table column or any value in an XML element, or attributes, are some examples.
2. Structure-based matching: approaches that take into account the structure in which data are represented, like an XML tree, an ontology graph, or a database tuple. In this category, it is important to note that we consider proposals that compare data structure and in some cases data values, but that do not distinguish data content from structure, treating both as a single thing.
3. Mixed matching: approaches that take into account both content and structure in which data are represented, and distinguish structure from content, respecting the semantic of each other. They apply different strategies for calculating the similarity between data instances.

Figure 1 presents the taxonomy that describes the approaches in this survey. As stated before, this classification does not intend to define a strict taxonomy for the subject, but, instead, a way of organizing and structuring the related work described in this survey.

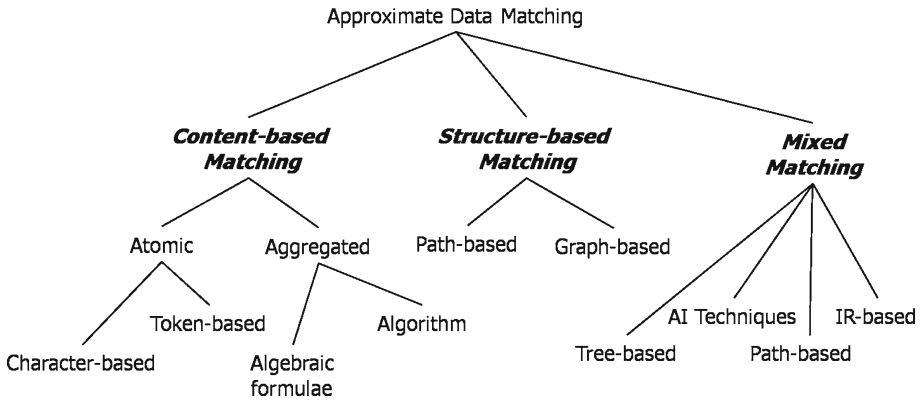

Fig. 1 Taxonomy of approximate data matching approaches

Content-based matching approaches are classified into those ones which compare instances based only on their *atomic values* and those ones which consider *aggregated values*. Comparison of atomic values usually applies token- or character-based similarity functions [33,74]. For aggregated value comparison, related work apply two basic approaches: (i) an *algebraic formula* to combine the individual similarity of the aggregated values through mathematic formulae; or (ii) an *algorithm* to execute a more complex computational method, such as a machine learning technique [9,8,87,88] or some other combination process [7,51].

The category *Structure-based matching* comprises works that considers structure and values as a single unit, including approaches based on graph-related methods [60,68,76] or path-based comparisons [16,91].

Mixed matching treats content and structure as separate issues, taking into account the semantics associated with the values by the structure wherein they are contained. It comprises work based on specialized tree-based [71,75], Artificial Intelligence (AI) techniques [66], path-based comparison [77], and Information Retrieval (IR) based approaches [18].

We give an overview of some approximate data matching approaches in the next sections, relating each work to its respective category in the taxonomy. Our intention is to provide a comprehensive state-of-art for the subject.

3 Content-based matching

In a content-based comparison, the way how two instances are compared in a matching process depends on the data values being compared. The comparison technique may calculate the similarity between atomic values, or measure the similarity between aggregated values. In the first case, a similarity function that sets a similarity score between the compared values is used. In the second case, individual attribute scores are combined in order to establish a final score. Combination functions or approaches based on AI techniques can be used here, like decision-trees or support vector machines (SVMs). Approaches related to these two categories are presented in the following.

3.1 Atomic values

Similarity functions used to compare atomic values can be organized into two main groups (not limited to them): (i) character-based, and (ii) token-based. The first one regards functions

that execute a character-to-character comparison, i.e., the analysis is performed on each individual character. Examples of such functions are *Levenshtein()*, *Jaro()* and *NGrams()*. Token-based functions execute a token-to-token comparison, i.e., analyze each token individually in a string. A token can be a word or any substring in a sentence. Examples of such functions are *SoftTFIDF()*, *MongeElkan()*, and *Jaccard()*.

There is an extensive literature related to similarity functions to compare textual values [19,32,33,74,87]. Implementations of these functions can be found in Java packages like SimMetrics [19] and SecondString [32]. These similarity functions, or a combination of them, are widely used in most approaches that have some atomic values comparison [8,9, 28,29,33,34,40,62,87,94]. Some of them [8,9] propose a unification of the field values in a record, applying a similarity function over this unification. Those works usually define a function based on some machine learning technique.

Similarity functions applied to atomic values usually depend on the value domain. Functions that work on people names and handle common writing heterogeneities are an example. Differences in name and surname order are very common, like "Kofi Annan" and "Annan, Kofi", as well as the presence, or absence, of the middle name ("Kofi Atta Annan"). At the same time, the accuracy when comparing two atomic values is strongly related to the chosen similarity function. A similarity function designed for people names being used in the domain of articles titles will certainly not generate satisfactory results.

There are extensive and comprehensive [33,64,85] reviews regarding similarity functions to compare textual values. Therefore, they will not be discussed in detail in this survey.

3.2 Aggregated values

The manipulation of aggregated values, such as relational tuples and XML trees, can be dependent on the application area, like data integration and similarity join queries, among others. We organize the related work based on how the individual attribute scores (obtained through similarity functions for atomic values) are combined, e.g., using (i) algebraic formulae; or (ii) algorithms.

3.2.1 Algebraic formulaes

Some approaches introduce proposals to combine the individual score values obtained for a set of attributes using specific metrics [21,39,51,73,82,103]. These metrics combine, mathematically, the similarity score reached for each attribute value, providing a single final similarity value between the aggregated values.

The Euclidean distance [79] is used by some approaches to combine distance scores generated for the attributes of a tuple. The scores of the individual attributes are calculated using a function for atomic value for each one, and after combined through the Euclidean metric.

The vectorial model [80] is redefined to allow the inclusion of an approximate operator (to be used instead of the equality operator) in the VAGUE System [73]. As in the traditional vectorial model, queries and the stored data are represented by means of vectors. However, the comparison between vectors is executed by an approximate operator. Basically, VAGUE applies individual distance functions to derive, from two vectors, a vector of their distances. Then, it weights the individual distances and maps the vector into a single distance by computing its Euclidean distance.

The approach followed by the WHIRL language [31] adopts a statistical similarity metric (cosine similarity metric) commonly used by the Information Retrieval (IR) community to compare documents. WHIRL is a logical language, based on probabilistic algebra, proposed

to database integration. The language was built to access STIR relations (Storing Text In Relations), where table attributes are text fragments, which justify the use of IR techniques. Each fragment defines a vector where each index represents the most important terms present in that fragment. The WHIRL language is used in different applications that implement the concept of textual similarity [29,30,34].

The use of Euclidean metric for combining the individual scores are useful for flat structures. However, the combination of atomic similarity scores in nested structures needs specific metrics. Dorneles et al. [40] propose an approach for the use of similarity metrics when querying XML databases. Taking into account that XML elements may be *complex*, i.e., they may contain other elements as values, the authors define similarity metrics for complex values, classified in two classes of metrics (*tuple* and *collection*—set and list) which are dependent on the type of structure of the complex element.

3.2.2 Algorithms

The work that propose specific algorithms for instance matching are constructed using some AI techniques [8,9,54,70,84,87,105], as well as other strategies such as association rules [38], ranking merging [51] or clustering techniques [10,23]. These approaches are presented as follows.

AI techniques Most of the proposals use some Artificial Intelligence technique, mainly machine learning. An example is the use of SVM [11]) for data matching processes. The Multiply Adaptive Record Linkage with Induction (MARLIN) system [8,9] implements a classifier-based approach that uses a framework for detecting duplicates using trainable functions for textual similarity (atomic values similarity), which are applied to each attribute of a tuple. Through a training process, these functions are adapted to each value domain of each attribute.

Another work that makes use of classifiers is described in [105], which combines multiple classifiers (technique widely used in data mining) to the identification of entities. For each entity attribute, different similarity functions are used. The classification techniques used in the experiments are the ones available in the *Weka* library,[1] like the 1-rule (1R), logistic regression and naive Bayes, among others.

In the *Active Atlas* system [87], learning rules are applied to a training process that uses *decision trees* [72]. It implements an approximate matching method that learns useful information to execute the mapping between two objects in order to integrate data sources. The Profile-Based Object Management (PROM) system [38] is also based on matching rules, implementing an approximate matching technique through the exploration of disjoint attributes, i.e., uncommon attributes between a pair of objects.

The MOMA system [89] proposes two mapping operators for approximate data matching: "merging mappings" and "mapping composition", combining the results of several matcher algorithms on both attribute values and contextual information.

Other proposals A proposal for merging ranked lists is described in [51], which adapts the well-known footrule distance [37]. The idea is to find a global ranking in which the distance between individual rankings can be minimized. The notion of distance between the rankings is based on the *Spearman Coefficient* [37]. To merge the similarity result scores and the rankings positions, the authors consider the selection problem of the *top-k* most relevant answers [24]. The intention is to derive a "best" final k-size ranking.

[1] http://wwww.cs.waikato.ac.nz/ml/weka.

Table 1 Related work on aggregated values Matching

Work	Combination process	Attribute comparison	User intervention	Attribute importance
Algebraic formulae				
Motro [73]	Euclidian metric	Domain functions	No	Weight
Cohen et al. [31]	Euclidian metric	NA	Yes	IR techniques
Yu et al. [103]	Euclidian metric	NA	No	NA
Dorneles et al. [40]	Different metrics	Domain functions	No	NA
Algorithms				
Tejada et al. [87]	Decision tree	Domain functions	Yes	Weight
Bilenko et al. [8]	SVM	Domain functions	Yes	NA
Doan et al. [38]	Matching rules	NA	Yes	Weight
Guha et al. [51]	Merge ranking	NA	Yes	NA
Chaudhuri et al. [23]	Clustering	NA	Yes	NA
Zhao and Ram [105]	SVM	Domain functions	Yes	NA
Puhlmann et al. [78]	SNM	NA	Yes	NA
Thor and Rahm [89]	Mapping operators	NA	Yes	NA

Clustering algorithms is another alternative to provide data matching focused on aggregated values [23]. The authors execute approximate matching for clustering purposes by considering the following properties: (i) tuples that duplicates in a group are closer to each other than to other tuples; and (ii) that their "local neighborhood" is empty or sparse. They use such properties to identify duplicate groups using clustering algorithms, proposing two criteria for that: the *Compact Set*, which captures the first property, and the *Sparse Neighborhood*, which works with the second property.

Puhlmann et al. [78] introduce an algorithm that is an adaptation of the Sorted Neighborhood Method (SNM) [53] in order to detect duplicate objects in XML documents. In the SNM method, key attributes are extracted, generating keys to the comparison, that are further ordered. Groups are defined for the ordered object keys, and the keys inside a group are compared with each other to detect duplicates.

3.3 Summary

Table 1 summarizes some important features related to the major work reviewed in this section: *Combination process* refers to the strategy to combine score values when object attributes are compared. *Attribute function* shows which functions (similarity/distance) are used to compare the attributes of an object. *User intervention* clarifies the need for intervention of an expert user prior or during the matching process. *Attributes importance* characterizes how the relevance of each attribute in the objects matching process is established.

We note, based on Table 1, a preference by Euclidean metric and AI techniques during the combination process as well as the application of domain-dependent functions to compare object attributes by some approaches. The definition of weights is taken into account by some works to define attribute importance during the matching processes.

4 Structure-based matching

Other major research line on approximate data matching analyzes the structure to compare data values, being strongly applied to semistructured data, usually XML documents. Such approaches consider structure and values as a single unit and do not take into account the semantics of the data. We organize the related work of this category into two main groups: path-based, where a comparison of one or more existing paths in tree-structured instances is performed; and graph-based, in which the methods applied to graph-based data structures are used to analyze the similarity between instances.

Path-based methods analyze a document structure (with or without its data content) and generate, through some process, unique identifications (IDs) to the several parts that compose its structure. These *IDs* are then compared between two documents in order to establish a similarity score. Graph-based methods establish a series of operations in order to transform one document hierarchical structure into another, by removing, inserting, moving, and renaming nodes from their graph structure. They basically apply the *edit-distance* concept from strings to trees [41].

4.1 Path-based matching

The work of Buttler [16] uses the concept of *shingles* defined by Broder [12] to compare the structure of XML documents. *Shingles* are extracted parts of a text block. After establishing the *shingles* for two given text blocks, their similarity is calculated by the relation between their intersection set size over their union set size. Buttler defines shingles representing the structure of XML documents as follows: for each node in an XML document, its path is converted into a value by applying a *hash function* over it, and this value represents a shingle.

Flesca et al. [43] introduce an approach that uses the concept of *Temporal Series* to compare the structure of two XML documents. In order to accomplish the comparison, a time series of values is established for each document, using certain encoding techniques.

Once the series are defined, a *discrete fourier transform* is applied over them, which transport the series from the domain of time to the domain of frequency. After this transformation, the series are compared. Since the same encoding is used for all compared documents, their series in the frequency domain can be directly compared in order to establish those which are more similar. The applied encodings respect the structure and elements relationships (hierarchical relationships), producing comparison results with quality similar to tree-edit distance approaches, which have a much higher cost of execution (See Sect. 4.2).

Vinson et al. [91] introduce a similarity function for XML paths called *PathSim*. This similarity function adapts a traditional edit-distance function, designed for string data, the *Levenstein* function [65], to compare XML paths. Their comparison method provides better results than existing methods, specially in multi-domain queries, where paths from various domains have to be compared.

4.2 Graph-based matching

The proposal of Nierman et al. [76] applies *tree-edit-distance* between trees representing the structures of XML documents as a measure for their structural similarity. It considers the moving of subtrees as part of the edit distance possible operations. According to the authors, exclusion and inclusion of subtrees better indicate the similarity between the dynamic

Table 2 Related work on structured-based matching

Work	Combination process	User intervention	Process objective
Path-based			
Buttler [16]	Shingles	No	Similarity value
Flesca et al. [43]	Temporal series	No	Similarity value
Vinson et al. [91]	Adaptated Levenstein function	No	Similarity value
Graph-based			
Nierman and Jagadish [76]	Adaptated tree-edit distance	No	Similarity value
Kailing et al. [60]	Specific formulae	No	Performance improvement
Chawathe et al. [25]	Edge-cover	No	Similarity value
Melnik et al. [68]	Similarity flooding	No	Nodes matching

structures of XML documents. It also establishes a set of allowed operations, reducing the complexity of the comparison, without impacting too much on the quality.

Kailing et al. [60] introduce a set of filters that allows, at low cost, to define if it is advantageous to compare two trees through *tree-edit-distance*. These filters are based on aspects such as tree height, node degrees, and node labels. Such filters can improve considerably the performance, given that *tree-edit-distance* is a NP-Complete problem for unordered trees[60].

Chawathe et al. [25] compare the structure of two XML documents through a technique known as *edge cover*. It establishes a bipartite graph where each half represents the tree of one of the documents compared. For each node in one of the graph halves, an operation is defined in order to convert it into a node from the other half. An edge is generated for each two connected nodes, labeled with the necessary operation in order to transform one node into the other. Once all the possible edges to transform the nodes from one half into the nodes from the other half are established, an algorithm calculates the set of edges that connects all the nodes between the two graphs at the lowest possible cost.

Melnik et al. [68] introduces a concept called "similarity flooding" in order to estipulate the similarity between nodes from two graphs with labeled edges. It is primarily used to elaborate mappings between two data schemas, but can be applied to other situations where identifying equivalent nodes between two graphs is necessary. Its basic idea is that nodes that are similar tend to have similar related nodes. For example, if a node "a" in a graph is similar to a node "b" in another graph (based on their contents through string similarity, for example), nodes connected to these nodes through edges with the same label tend to be similar. For example, if a node "a1" is connected to "a" through an edge labeled "z" and there is a node "b1" connected to "b" through an edge also labeled "z", the possibility of "a1" and "b1" being similar is greater. Through an iterative procedure, the "similarity flooding" algorithm weights these similarity probabilities and establishes mapping between the nodes from the two graphs.

4.3 Summary

Table 2 summarizes the most important aspects related to the work in the Structured-based Matching category. The first two items are the same of Table 1. As shown in Table 2, all approaches here require no user intervention, i.e., they are all automatic. The item *Process Objective* means whether the process executes the comparison or improves existing comparison processes.

5 Mixed matching

This category comprises work that considers both structure and values of data instances in the matching process. The structure is analyzed in order to define which data values should be compared. On considering proposals which distinguish between structure and values, we can categorize them into four groups: tree-based, in which extended versions of the tree-edit-distance and metrics that compose similarity scores over tree-based structures; AI Techniques, where techniques such as clustering methods and Inverse Document Frequency (IDF) are applied to graph-based data structures; path-based, in which tree-based data comparison that considers values and associated paths to these values; and finally, IR-based, where we consider those work that are based on IR techniques.

5.1 Tree-based matching

Regarding *tree-based* approaches, related work focus on XML documents. They are well suited to small volumes of data, given the fact that their methods require the comparison through string-related metrics between pairs of data.

Milano et al. [71] introduce a method to estimate the edit-distance between XML documents that considers the semantics of the elements. It calculates the edit-distance between data contained under the same path (which provide a semantic context for the element) and proceeds a bottom-up combination of these in order to establish the edit-distance for complex elements.

Ma et al. [67] introduce a method to define the similarity between XML documents under a common structure. The similarity between two elements is defined through the average of its attributes and textual similarity. The similarity between complex elements is defined as the similarity between their sub-elements, identified by the same tag. It introduces a way to deal with the comparison of value collections in an asymmetric way, which takes into account the containment aspect between the two collections.

5.2 AI techniques

Approaches based on AI Techniques also focus on XML data. Yang et al. [101], for example, calculate the similarity between documents by their representations in the Structured Linked Vector Model Latent Semantic Indexing (SLVM-LSI), which is their extension to the Structured Linked Vector Model (SLVM). Latent Semantic Indexing [35] is a technique used in IR for solving problems such as synonyms and polyssemies. On combining these techniques, Yang et al. demonstrate that they are able to include documents' semantics in a clustering process and present better clustering results if compared with the solely application of SLVM. The work also introduces an iterative learning method to calculate the similarity between documents, different from other approaches that use heuristic algorithms.

In [66], the authors propose a method for fuzzy duplicate detection in hierarchical and semistructured XML data. They consider not only the duplication of a complex element hierarchical structure, as related work do, but also the probability of descendant elements being duplicates, which is computed using a Bayesian network.

5.3 Path-based matching

The work of Park et al. [77] follows an specific approach, that we called *Path-based matching*. It introduces a system to search XML documents that contain similar data under a given context. To define the context, data to be searched have a path in XML associated to it. During

the search, similar paths to the given one are detected, taking into account the semantics of the elements. This allows approximate data searches in XML documents, resulting in more relevant information. In [90], the similarity values produced from structure comparison and content comparison are associated with weights to measure the overall document similarity.

5.4 IR-based matching

IR-based approaches work on large data volumes through suitable metrics. They do not require user intervention during the process (which would be unworkable, due to the amount of data), but allow previous user parametrization or data adaptation to provide better results. They focus on similarity, either to correlate similar data or to detect duplicates for cleaning purposes.

Carvalho et al. [18] propose a method to identify similar objects from multiple Web data sources. By analyzing and transforming the structure and data of compared objects into vectors, the approach allows IR-related methods (like the vectorial model) to be used in order to compare those objects. It supports different methods to compare the data-related vectors, providing different techniques to deal with inconsistent properties between the objects of the sources.

Weis et al. [95] detect duplicate objects in XML documents. Its solution is based on a iterative top-down analysis of the XML document objects, detecting duplicates on each level of the document tree. It compares object data tokens (words, numbers, etc.) first by the application of string-similarity functions. This reduces equivalent tokens to a same token. Once this process is over, the token sets between the objects are compared by applying a variation of the *IDF* metric from the IR-field and equivalent objects are replaced by a single object.

Weis et al. [96] propose a framework to detect duplicate objects in XML documents. This framework consists of three major components: Candidate Definition, Duplicate Definition and Duplicate Detection. In *Candidate Definition*, Object Descriptions (ODs) are defined. Object Descriptions (ODs) represent objects in XML documents, as well as the relations and dependencies between them. Duplicate Definition establishes when two *ODs* represent the same object. Duplicate Detection represents the process of comparing generated *ODs* from the XML documents and identifying the duplicates.

A new method to detect duplicates in XML documents is introduced in Weis et al. [97]. It considers complex relations such as hierarchical relations with multiple cardinality and novels by establishing an order to compare the objects in XML documents. This order reduces the number of necessary reclassifications for some elements due to newfound similarities for elements which they depend upon.

5.5 Summary

Table 3 summarizes the most important features for the work that fit in this category, which are the metric used in the process and the fact that the method needs the user intervention.

The trend here is for string similarity metrics and automatic approaches, except for the IR-based category, which provides heterogeneous metrics and usually requires user intervention.

6 Application areas

Approximate data matching can be applied to several application areas. The most relevant ones are presented in the following:

Table 3 Related work on mixed matching

Work	Metric	User intervention
Tree-based		
Milano et al. [71]	String similarity	No
Ma and Chbeir [67]	String similarity	No
Al techniques		
Yang et al. [101]	SLVM-LSI	No
Luis et al. [66]	String similarity	No
Path-based		
Park et al. [77]	String similarity	No
IR-based		
Carvalho and da Silva [18]	Vector similarity	Yes
Weis and Naumann [95]	IDF	No
Weis and Naumann [96]	Dogmatix	Yes
Weis and Naumann [97]	Sorted neighborhood	Yes

6.1 Data integration

In a general way, various problems are dealt by data integration solutions, such as finding the same object represented in different ways in data sources being integrated [87], construction of wrappers to automatically integrate sources [17], and generation of plans to integrate data coming from the sources [61,63,69], among others. Despite these different approaches, the general purpose of data integration proposals is to generate, at the end of the process, mappings that indicate which pair of objects from different sources correspond to the same real-world object.

There is extensive literature related to data integration. Besides the proposals introduced above, there are other relevant ones. In [54,55], the authors describe a framework to combine values coming from different sources. In this framework, an heuristic *join* operator is defined, which has a set of parameters that are used during the integration process. The Citeseer projet [10,48] is an IR system that is able to locate articles automatically, to extract citations, and to identify citations for the same article that occur in different formats, as well as to identify the context of the citations in the article body.

In [81,82], operators for similarity joining and similarity clustering are proposed for data integration systems. The authors describe the semantics and implementation of these operators, as well as optimization techniques during the integration process. In [52], a method called *Multi Pass Neighborhood Method* is proposed to detect the maximum number of exact and approximate duplicate records in the shortest amount of time. Such detection is performed through a set of rules.

Besides these works, there are those which introduce comparison methods [8,18,25,26,60,76] to support data integration systems. These work base their approaches on various methods, where tree-edit-distance variations and IR-vector-based comparisons are included.

6.2 Similarity queries

Approaches have been proposed for approximate matching in the context of query languages with operators that process the comparison of values by approximation [1,4,6,13,15,24,73,

83,99]. In this scenario, it is common to find proposals which use a similarity operator as the basis of a join implementation using various similarity functions.

One of these proposals is the WHIRL language [50], that executes textual joins in a DBMS without any modification in the query processor. The join operator uses the *cosine* similarity function [80]. A join is executed with the support of Web Services that extract and materialize the data from different sources in a relational database, solving inconsistencies in the data values.

Other class of work relates to query processing with similarity functions consideration. Some of them propose extensions to the SQL language [46,56,58,82], algebras with support for similarity [2,36,57], similarity operators [3,27,49,50,59,100], and approximate searches in XML documents [77,91]. A comprehensive study about similarity-based queries and related issues can be found in [85].

6.3 Data cleaning

The main purpose of data cleaning systems is to detect and correct errors in data coming from external sources that are going to be stored in a data warehouse. Such data may contain inconsistencies like different conventions, missing fields, typographic errors, typing errors, and grammatical errors, which must be dealt before data being stored in a data warehouse. Some proposals focus on solving such a problem [8,9,20,22,51,62].

In [21], for instance, a fuzzy algorithm is proposed. It extends the Levenstein distance to be applied to a record. The proposal introduced in [44,45] suggests the use of a matching operator that basically calculates the distance value between each pair of tuples from a cartesian product, using an arbitrary distance function.

Recent works have been focusing on data cleaning in XML documents. They use various approaches, including tree-based methods [67,71], AI techniques [66,78,96,97], and IR-field-like methods [95].

6.4 Clustering

In clustering systems, the idea is to establish a reduced view of existing data, capturing their most important features, and storing such data. The stored data can then be queried quickly based on these relevant features.

In XML clustering systems, such as [16,43], the clustering criteria are based solely on the tree-structure aspects of the data. Recent work, such as [90,101,102], have been considering both structure and data in the clustering process. The proposal in [101] analyzes how data contained inside XML documents are structured and how they relate to other close data. In [101], a similarity model is proposed in order to measure similarities scores between XML documents, which combines structure and contents of the XML documents. Based on this similarity model, the authors adopt hierarchy clustering algorithm to cluster XML documents. In [90], the similarity values produced from structure comparison and content comparison are combined with weights to measure the overall document similarity.

7 Overall comparison

Tables 4 and 5 provide an overall comparison of the main related work, considering relevant features that were analyzed in this survey.

Table 4 Comparative analysis (I)

Work	Nomenclature	Application area	Data volume	Data nature
Content matching				
Yu et al. [103]	Top k-queries	Similarity queries	Medium	Database tuples
Motro [73]	Vague queries	Similarity queries	Medium	Database tuples
Cohen et al. [31]	Hardening soft DB	Data cleaning	Medium	Text fragments
Guha et al. [51]	NA	Data cleaning	Small	Ranked list
Tejada et al. [87]	Object identification	Data integration	Medium	Data records
Bilenko et al. [8]	Record linkage	Data cleaning	Small	Database tuples
Zhao and Ram [105]	Entity identification	Data integration	Small	Database tuples
Doan et al. [38]	Object matching	Data integration	Small	Objects
Thor and Rahm [89]	Object matching	Data integration	Medium	Objects
Chaudhuri et al. [23]	Duplicate elimination	Data cleaning	Medium	Database tuples
Puhlmann et al. [78]	Duplicate detection	Data cleaning	Medium	XML
Structure matching				
Buttler [16]	Structural similarity	Clustering	Large	XML
Flesca et al. [43]	Structural similarity	Clustering	Large	XML
Vinson et al. [91]	Path matching	Similarity queries	Small	XML
Nierman [76]	Structural similarity	Similarity queries	Small	XML
Kailing et al. [60]	Similarity search	Similarity queries	Large	Semi-structured
Chawathe et al. [25]	Change detection	Similarity queries	Small	Semi-structured
Melnik et al. [68]	Graph matching	Data integration	Small	Semi-structured
Mixed matching				
Milano et al. [71]	NA	Similarity queries	Small	XML
Ma and Chbeir [67]	XML document	Similarity queries	Small	XML
Yang et al. [101]	XML document	Clustering	Large	XML
Luis et al.[66]	Duplicate detection	Data cleaning	Small	XML
Park et al. [77]	XML document	Similarity queries	Medium	XML
Carvalho et al. [18]	Object identification	Data cleaning	Medium	Semi-structured
Weis and Naumann [95]	Duplicate detection	Data cleaning	Large	XML
Weis and Naumann [96]	Duplicate detection	Data cleaning	Large	XML
Weis and Naumann [97]	Duplicate detection	Data cleaning	Large	XML

We highlight in Table 4 that the *nomenclature* adopted by the research community is not unique when the problem is to identify various representations of a same real-world object. Sometimes, it is related to the considered application area, like *duplicate detection* and *duplicate elimination* for the *data cleaning* area. Despite that, the intended result is usually a set of similar objects, which can be ranked by the highest affinity scores.

In terms of *application areas*, we see many solutions for *data cleaning*, given the (not new) research on management of data warehouses. We have also several proposals for *similarity queries*, which is an important issue, with applications, for example, in mediation-based systems and Web data searching, that demand access to heterogeneous data sources. Many approaches focus on XML data (see *data nature*), considering the availability of data sources in this format, as well as the conversion of database or application data to XML for

Table 5 Comparative analysis (II)

Work	Attributes used	Preprocessing	Adaptability	Similarity/ distance
Content matching				
Yu et al. [103]	NA	NA	NA	Distance
Motro [73]	Common	NA	NA	Distance
Cohen et al. [31]	Common	NA	NA	Similarity
Guha et al. [51]	Common and uncommon	NA	NA	NA
Tejada et al. [87]	Common	NA	NA	Similarity
Bilenko et al. [8]	Common	NA	NA	Similarity
Zhao and Ram [105]	NA	NA	NA	NA
Doan et al. [38]	Common and uncommon	NA	NA	Similarity
Thor and Rahm [89]	Common	NA	NA	NA
Chaudhuri et al. [23]	Common	NA	NA	NA
Puhlmann et al. [78]	Common	Yes	Attributes	Similarity
Structure matching				
Buttler [16]	NA	No	NA	Similarity
Flesca et al. [43]	NA	No	Encoding	Similarity
Vinson et al. [91]	NA	No	NA	Similarity
Nierman [76]	Common and uncommon	No	NA	Distance
Kailing et al. [60]	NA	No	NA	NA
Chawathe et al. [25]	Common and uncommon	No	NA	Distance
Melnik et al. [68]	NA	No	NA	Similarity
Mixed matching				
Ma and Chbeir [71]	Common	No	NA	Distance
Ma and Chbeir [67]	Common and uncommon	No	NA	Similarity
Yang et al. [101]	Common and uncommon	No	Learning	Similarity
Luis et al.[66]	Common	No	Learning	Similarity
Park et al. [77]	NA	No	NA	Similarity
Carvalho et al. [18]	Common and uncommon	No	Comparison	Similarity
Weis and Naumann [95]	Common and uncommon	Yes	NA	Similarity
Weis and Naumann [96]	Common and uncommon	Yes	OD Definition	Similarity
Weis and Naumann [97]	Common and uncommon	Yes	OD Definition	Similarity

interchanging purposes. A little bit different is the *data volume* managed by the approaches, which shows a balance between small, medium, and large data sets. Such a situation demands an enhancement of several proposals in order to be more scalable.

In Table 5, on *used attributes*, we see a tendency for approaches that consider the common attributes of a data object in order to define a similarity score, given that such attributes provide better hints for assuming that two objects have a semantic affinity. However, the additional consideration of the uncommon ones could be useful to improve the similarity accuracy, in terms, for example, of the semantic context of the object. A trend also for no data *pre-processing* seems to be positive because reduces the overhead in the data processing in order to define the approximate matchings.

Finally, we show that many approaches use *similarity functions*, or adaptations of well-known ones, to provide specific solutions for their specific data contents. On the other hand, the *adaptability* of the approaches to deal with the continuous matching of new data samples is not expressive, being considered mainly in application areas that apply AI techniques, like *clustering* and *data cleaning*.

8 Concluding Remarks

This paper presents a survey about *approximate data matching*, which is an active research issue with several applications in the area of data management. We propose a taxonomy that is able to categorize the existing related work. This taxonomy is centered in two axes: *content matching* and *structure matching*. The first one relates to works that analyze only the content of data in order to provide the matching, while the second one comprises works that consider data structure in their matching solutions. Approaches on both categories are detailed and compared, providing a comprehensive analysis of the related work. The comparative analysis, in particular, highlights some tendencies or aspects where research is more concentrated on.

Future work includes the investigation of new application areas where data matching have been used as well as a comparison of related work with regard to performance issues.

Acknowledgements This work was partially supported by the CNPq Brazilian research support foundation in the scope of the Digitex Project. CTInfo Process Nr.: 550.845/2005-4.

References

1. Agrawal R, Faloutsos C, Swami AN (1993) Efficient similarity search in sequence databases. In: Proceedings of the 4th international conference on foundations of data organization and algorithms (FODO). Springer, London, pp 69–84
2. Al-Khalifa S, Yu C, Jagadish HV (2003) Querying structured text in an xml database. In: Proceedings of the 29th ACM SIGMOD international conference on management of data (SIGMOD). ACM, New York, pp 4–15
3. Arasu A, Chaudhuri S, Kaushik R (2008) Transformation-based framework for record matching. In: Proceedings of the 2008 IEEE 24th international conference on data engineering (ICDE). IEEE Computer Society, Washington, pp 40–49
4. Arasu A, Ganti V, Kaushik R (2006) Efficient exact set-similarity joins. In: Proceedings of the 32nd international conference on very large data bases (VLDB). VLDB Endowment, pp 918–929
5. Aygun RS (2008) S2s: structural-to-syntactic matching similar documents. Knowl Inf Syst 16(3):303–329
6. Barioni MC, Razente HL, Traina C Jr, Traina AJM (2005) Querying complex objects by similarity in sql. In: Proceedings of the 20th Brazilian symposium on databases (SBBD), pp 130–144
7. Bhattacharya I, Getoor L (2005) Relational clustering for multi-type entity resolution. In: Proceedings of the 4th international workshop on multi-relational mining (MRDM). ACM, New York, pp 3–12
8. Bilenko M, Mooney R, Cohen W, Ravikumar P, Fienberg S (2003) Adaptive name matching in information integration. IEEE Intell Syst 18(5):16–23
9. Bilenko M, Mooney RJ (2003) Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, New York, pp 39–48
10. Bollacker K, Lawrence S, Giles CL (1998) CiteSeer: an autonomous web agent for automatic retrieval and identification of interesting publications. In: Proceedings of the 2nd international conference on autonomous agents. ACM Press, New York, pp 116–123

11. Boser BE, Guyon IM, Vapnik V (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on computational learning theory (COLT). ACM, New York, pp 144–152
12. Broder A (1997) On the resemblance and containment of documents. In: Proceedings of the compression and complexity of sequences (SEQUENCES). IEEE Computer Society, Washington, p 21
13. Bruno N, Chaudhuri S, Gravano L (2002) Top-k selection queries over relational databases: mapping strategies and performance evaluation. ACM Trans Database Syst 27(2):153–187
14. Bryan B, Schneider J, Nichol R, Miller C, Genovese C, Wasserman L (2006) Active learning for identifying function threshold boundaries. Advances in neural information processing systems (NIPS). MIT Press
15. Bueno R, Traina AJM, Traina C Jr (2005) Accelerating approximate similarity queries using genetic algorithms. In: Proceedings of the 2005 ACM symposium on applied computing (SAC), Santa Fe, New Mexico, USA, pp 617–622
16. Buttler D (2004) A short survey of document structure similarity algorithms. In: Proceedings of the international conference on internet computing, Las Vegas, Nevada, USA, pp 3–9
17. Carman MJ, Knoblock CA (2005) Inducing source descriptions for automated web service composition. In: Proceedings of the AAAI workshop on exploring planning and scheduling for web services, grid, and autonomic computing, Pittsburgh, Pennsylvania
18. Carvalho JCP, da Silva AS (2003) Finding similar identities among objects from multiple web sources. In: Proceedings of the 5th ACM international workshop on web information and data management (WIDM). ACM, New York, pp 90–93
19. Chapman S (2004) SimMetrics: a Java & C# .NET library of similarity metrics. http://sourceforge.net/projects/simmetrics/. Accessed 13 March 2009
20. Chaudhuri S, Chen B-C, Ganti V, Kaushik R (2007) Example-driven design of efficient record matching queries. In: Proceedings of the 33rd international conference on very large data bases (VLDB), VLDB Endowment, pp 327–338
21. Chaudhuri S, Ganjam K, Ganti V, Motwani R (2003) Robust and efficient fuzzy match for online data cleaning. In: Proceedings of the 29th ACM SIGMOD international conference on management of data (SIGMOD). ACM, New York, pp 313–324
22. Chaudhuri S, Ganti V, Kaushik R (2006) A primitive operator for similarity joins in data cleaning. In: Proceedings of the 22nd international conference on data engineering (ICDE). IEEE Computer Society, Washington, p 5
23. Chaudhuri S, Ganti V, Motwani R (2005) Robust identification of fuzzy duplicates. In: Proceedings of the 21st international conference on data engineering (ICDE). IEEE Computer Society, Washington, pp 865–876
24. Chaudhuri S, Gravano L (1999) Evaluating top-k selection queries. In: Proceedings of the 25th international conference on very large data bases (VLDB). Morgan Kaufmann Publishers Inc., San Francisco, pp 397–410
25. Chawathe SS, Garcia-Molina H (1997) Meaningful change detection in structured data. In: Proceedings of the 1997 ACM SIGMOD international conference on management of data (SIGMOD). ACM, New York, pp 26–37
26. Cheng T, Chang KC-C (2007) Entity search engine: towards agile best-effort information integration over the web. In: Third biennial conference on innovative data systems research (CIDR), Asilomar, CA, USA, pp 108–113
27. Chuan Xiao, Wei Wang XL (2008) Ed-join: an efficient algorithm for similarity joins with edit distance constraints. In: Proceedings of the VLDB Endow. VLDB Endowment, pp 933–944
28. Cohen WW (1998a) Integration of heterogeneous databases without common domains using queries based on textual similarity, pp 201–212
29. Cohen WW (1998b) Providing database-like access to the web using queries based on textual similarity. SIGMOD Rec 27(2):558–560
30. Cohen WW (1999) Recognizing structure in web pages using similarity queries. In: Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference innovative applications of artificial intelligence (AAAI/IAAI). American Association for Artificial Intelligence, Menlo Park, pp 59–66
31. Cohen WW (2000) Whirl: a word-based information representation language. Artif Intell 118(1–2): 163–196
32. Cohen WW, Ravikumar P, Fienberg S (2003a) Secondstring: open source java-based package of approximate string-matching. http://secondstring.sourceforge.net/. Accessed 20 Dec 2005
33. Cohen WW, Ravikumar P, Fienberg SE (2003b) A comparison of string distance metrics for name-matching tasks. In: Proceedings of workshop on information integration on the Web, IIWeb, pp 73–78

34. Cohen WW, Richman J (2002) Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, New York, pp 475–480
35. Deerwester SC, Dumais ST, Landauer TK, Furnas GW, Harshman RA (1990) Indexing by latent semantic analysis. JASIS 41(6):391–407
36. Dey D, Sarkar S (1996) A probabilistic relational model and algebra. ACM Trans Database Syst 21(3):339–369
37. Diaconis P, Graham R (1997) Spearman's footrule as a measure of disarray. J R Stat Soc 39(2):262–268
38. Doan A, Lu Y, Lee Y, Han J (2003) Profile-based object matching for information integration. IEEE Intell Syst 18(5):54–59
39. Dorneles CF, Heuser CA, da Silva AS, de Moura ES (2009) A generic strategy for combining similarity metrics in approximate matching. Inf Syst (Oxford) 34:673–689
40. Dorneles CF, Heuser CA, Lima AEN, da Silva AS, de Moura ES (2004) Measuring similarity between collection of values. In: Proceedings of the 6th annual ACM international workshop on web information and data management (WIDM). ACM, New York, pp 56–63
41. Dulucq S, Touzet H (2003) Analysis of tree edit distance algorithms. In: Proceedings of the 14th annual symposium combinatorial pattern matching (CPM), pp 83–95
42. Elmagarmid AK, Ipeirotis PG, Verykios VS (2007) Duplicate record detection: a survey. IEEE Trans Knowl Data Eng 19:1–16
43. Flesca S, Manco G, Masciari E, Pontieri L, Pugliese A (2005) Fast detection of XML structural similarity. IEEE Trans Knowl Data Eng 17(2):160–175
44. Galhardas H, Florescu D, Shasha D, Simon E (2000) An extensible framework for data cleaning. In: Proceedings of the 16th international conference on data engineering (ICDE). IEEE Computer Society, Washington, p 312
45. Galhardas H, Florescu D, Shasha D, Simon E, Saita C-A (2001) Declarative data cleaning: language, model, and algorithms. In: Proceedings of the 27th international conference on very large data bases (VLDB). Morgan Kaufmann Publishers Inc., San Francisco, pp 371–380
46. Gao L, Wang M, Wang XS, Padmanabhan S (2004) Expressing and optimizing similarity-based queries in sql. In: 23rd international conference on conceptual modeling (ER), Shanghai, China, pp 464–478
47. Getoor L, Taskar B (eds) (2007) Introduction to statistical relational learning. MIT Press, Cambridge
48. Giles CL, Bollacker K, Lawrence S (1998) CiteSeer: an automatic citation indexing system. In: Proceedings of the third ACM conference on digital libraries (DL). ACM, New York, pp 89–98
49. Gravano L, Ipeirotis PG, Jagadish HV, Koudas N, Muthukrishnan S, Srivastava D (2001) Approximate string joins in a database (almost) for free. In: Proceedings of the 27th international conference on very large data bases (VLDB). Morgan Kaufmann Publishers Inc., San Francisco, pp 491–500
50. Gravano L, Ipeirotis PG, Koudas N, Srivastava D (2003) Text joins in an rdbms for web data integration. In: Proceedings of the 12th international conference on World Wide Web (WWW). ACM, New York, pp 90–101
51. Guha S, Koudas N, Marathe A, Srivastava D (2004) Merging the results of approximate match operations. In: Proceedings of the thirtieth international conference on very large data bases (VLDB). VLDB Endowment, pp 636–647
52. Hernandez MA, Stolfo SJ (1995) The merge/purge problem for large databases. SIGMOD Rec 24(2):127–138
53. Hernández MA, Stolfo SJ (1998) Real-world data is dirty: data cleansing and the merge/purge problem. Data Min Knowl Discov 2(1):9–37
54. Huffman S, Steier D (1995a) Heuristic joins to integrate structured heterogeneous data. In: Working notes of the AAAI spring symposium on information gathering from heterogeneous, distributed environment, pp 74–77
55. Huffman SB, Steier D (1995b) A navigation assistant for data source selection and integration, Working papers, Price Waterhouse
56. Ilyas F, Aref G, Elmagarmid K (2004) Supporting top-k join queries in relational databases. VLDB J 13(3):754–765
57. Ilyas IF, Aref WG (2005) Rank-aware query processing and optimization. In: Proceedings of the 21st international conference on data engineering (ICDE). IEEE Computer Society, Washington, p 1144
58. Ilyas IF, Aref WG, Elmagarmid AK (2003) Supporting top-k join queries in relational databases. In: VLDB
59. Jin L, Li C, Mehrotra S (2002) Efficient similarity string joins in large data sets, technical report, University of California, Irvine. http://www.ics.uci.edu/chenli/pub/strjoin.ps

60. Kailing K, Kriegel H-P, Schönauer S, Seidl T (2004) Efficient similarity search for hierarchical data in large databases. In: 9th international conference on extending database technology (EDBT), Heraklion, Crete, Greece, pp 676–693
61. Knoblock CA, Ambite JL, Thakkar S (2003) A view integration approach to dynamic composition of web services. In: Proceedings of ICAPS workshop on planning for web services
62. Koudas N, Marathe A, Srivastava D (2004) Flexible string matching against large databases in practice. In: Proceedings of the thirtieth international conference on very large data bases (VLDB), VLDB Endowment, pp 1078–1086
63. Laender AHF, Gonalves MA, Roberto PA (2004) Bdbcomp: building a digital library for the brazilian computer science community. In: ACM/IEEE joint conference on digital libraries (JCDL), Tuscon, AZ, USA, pp 23–24
64. Lee L (2001) On the effectiveness of the skew divergence for statistical language analysis. Artif Intell Stat, pp 65–72
65. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. Soviet Phys Doklady, vol 10
66. Luis Leit a, Calado P, Weis M (2007) Structure-based inference of xml similarity for fuzzy duplicate detection. In: Proceedings of the sixteenth ACM conference on information and knowledge management (CIKM). ACM, New York, pp 293–302
67. Ma Y, Chbeir R (2005) Content and structure based approach for xml similarity. In: Proceedings of the fifth international conference on computer and information technology (CIT). IEEE Computer Society, Washington, pp 136–140
68. Melnik S, Garcia-Molina H, Rahm E (2002) Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th international conference on data engineering (ICDE). IEEE Computer Society, Washington, pp 117–128
69. Michalowski M, Ambite JL, Knoblock CA, Minton S, Thakkar S, Tuchinda R (2004) Retrieving and semantically integrating heterogeneous data from the web. IEEE Intell Syst 19(3)
70. Michalowski M, Thakkar S, Knoblock CA (2005) Automatically utilizing secondary sources to align information across data sources. AI Mag 26(1):72–79
71. Milano D, Scannapieco M, Catarci T (2006) Structure aware xml object identification. In: Proceedings of the first Int'l VLDB workshop on clean databases (CleanDB). Seoul, Korea
72. Mitchell TM (1997) Machine learning. McGraw-Hill, New York
73. Motro A (1988) Vague: a user interface to relational databases that permits vague queries. ACM Trans Off Inf Syst 6(3):187–214
74. Navarro G (2001) A guided tour of approximate string matching. ACM Comput Surv 33(1):31–88
75. Nayak R (2008) Fast and effective clustering of xml data using structural information. Knowl Inf Syst 14(2):197–215
76. Nierman A, Jagadish HV (2002) Evaluating structural similarity in XML documents. In: 5th international workshop on the web and databases (WebDB), Madison, Wisconsin, USA, pp 61–66
77. Park U, Seo Y (2005) An implementation of XML documents search system based on similarity in structure and semantics. In: International workshop on challenges in web information retrieval and integration
78. Puhlmann S, Weis M, Naumann F (2006) XML duplicate detection using sorted neighborhoods. In: Proceedings of the 10th international conference on extending database technology (EDBT), pp 773–791
79. Ragnemalm I (1993) The euclidean distance transform, Ph.D thesis, Department of Electrical Engineering, Linkpping University
80. Salton G, McGill M (1984) Introduction to modern information retrieval. McGraw-Hill, New York
81. Schallehn E, Sattler K-U, Saake G (2001) Advanced grouping and aggregation for data integration. In: CIKM '01 proceedings of the tenth international conference on information and knowledge management. ACM, New York, pp 547–549
82. Schallehn E, Sattler K-U, Saake G (2004) Efficient similarity-based operations for data integration. Data Knowl Eng 48(3):361–387
83. Shatkay H, Zdonik SB (1996) Approximate queries and representations for large data sequences. In: Proceedings of the twelfth international conference on data engineering (ICDE). IEEE Computer Society, Washington, pp 536–545
84. Shen W, Li X, Doan A (2005) Constraint-based entity matching. In: Proceedings of the 20th national conference on artificial intelligence (AAAI). AAAI Press, pp 862–867
85. Stasiu RK, Heuser CA, Silva R (2005) Estimating recall and precision for vague queries in databases. In: Proceedings of the 17th international conference advanced information systems engineering (CAISE), pp 187–200

86. Takasu A, Fukagawa D, Akutsu T (2007) Statistical learning algorithm for tree similarity. In: Proceedings of the seventh IEEE international conference on data mining (ICDM). IEEE Computer Society, Washington, pp 667–672
87. Tejada S, Knoblock CA, Minton S (2001) Learning object identification rules for information integration. Inf Syst 26(8)
88. Tejada S, Knoblock CA, Minton S (2002) Learning domain-independent string transformation weights for high accuracy object identification, pp 350–359
89. Thor A, Rahm E (2007) Moma—a mapping-based object matching system. In: 'Third biennial conference on innovative data systems research (CIDR). Asilomar, CA, USA, pp 247–258
90. Tran T, Nayak R, Bruza P (2008) Combining structure and content similarities for xml document clustering. In: Proceedings of the 7th Australasian data mining conference (AusDM), vol 87. ACS, Glenelg, pp 219–226
91. Vinson AR, Heuser CA, Silva AS, de Moura ES (2007) An approach to xml path matching. In: Proceedings of the 9th annual ACM international workshop on web information and data management (WIDM). ACM, New York, pp 17–24
92. Wan X (2008) Beyond topical similarity: a structural similarity measure for retrieving highly similar documents. Knowl Inf Syst 15(1):55–73
93. Wang T-Y, Ré C, Suciu D (2008) Implementing not exists predicates over a probabilistic database. In: Proceedings of the international workshop on quality in databases and management of uncertain data, pp 73–86
94. Wang W, Xiao C, Lin X, Zhang C (2009) Efficient approximate entity extraction with edit distance constraints. In: Proceedings of the 35th SIGMOD international conference on management of data (SIGMOD). ACM, New York, pp 759–770
95. Weis M, Naumann F (2004) Detecting duplicate objects in xml documents. In: Proceedings of the 2004 international workshop on information quality in information systems (IQIS). ACM, New York, pp 10–19
96. Weis M, Naumann F (2005) Dogmatix tracks down duplicates in xml. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data (SIGMOD). ACM, New York, pp 431–442
97. Weis M, Naumann F (2006) Detecting duplicates in complex xml data. In: Proceedings of the 22nd international conference on data engineering (ICDE). IEEE Computer Society, Washington, p 109
98. Widom J (2008) Managing and mining uncertain data. In: Trio: a system for data, uncertainty, and lineage. Springer, Berlin
99. Xiao C, Wang W, Lin X, Shang H (2009) Top-k set similarity joins. In: Proceedings of the IEEE international conference on data engineering (ICDE). IEEE Computer Society, Washington, pp 916–927
100. Xiao C, Wang W, Lin X, Yu JX (2008) Efficient similarity joins for near duplicate detection. In: Proceeding of the 17th international conference on World Wide Web (WWW). ACM, New York, pp 131–140
101. Yang J, Cheung WK, Chen X (2005) Integrating element and term semantics for similarity-based xml document clustering. In: Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence (WI). IEEE Computer Society, Washington, pp 222–228
102. Yongming G, Dehua C, Jiajin L (2008) Clustering xml documents by combining content and structure. In: Proceedings of the international symposium on information science and engieering. IEEE Computer Society, Los Alamitos, pp 583–587
103. Yu CT, Philip G, Meng W (2003) Distributed top-n query processing with possibly uncooperative local systems. In: Proceedings of the 29th international conference on very large data bases (VLDB). VLDB Endowment, pp 117–128
104. Zhao H (2008) Instance weighting versus threshold adjusting for cost-sensitive classification. Knowl Inf Syst 15(3):321–334
105. Zhao H, Ram S (2005) Entity identification for heterogeneous database integration: a multiple classifier system approach and empirical evaluation. Inf Syst 30(2):119–132

Author Biographies

Prof. Dr. Carina Friedrich Dorneles is a Professor in the Federal University of Santa Catarina (UFSC), Brazil. She held visiting post (visiting student) in the University of Washington, Seattle, USA during 2003. Her research interest are in database and data management areas, specifically in data integration using similarity function and XML data. Dr. Dorneles is a member of UFSC Research Group on Databases (GBD-UFSC), and has served as external reviewer for several national and international conferences and external reviewer of national and international journals.

Rodrigo Gonçalves Msc. is a system analyst and researcher in the Database area. Mr. Gonçalves got his Masters' Degree in Computer Science at Federal University of Santa Catarina, Brazil, in 2007, working on the XML instance matching subject. Msc. Gonçalves is a member of UFSC Research Group on Databases (GBD-UFSC), having publications in national and international Database conferences. His interest areas include XML data management and integration, as well as data matching.

Prof. Dr. Ronaldo dos Santos Mello is a Professor in the Federal University of Santa Catarina (UFSC), Brazil. He held a Post doc grant as a visiting professor in the University of Utah, Salt Lake City, USA during 2009. His research is concentrated in the Database area, with a current focus on data integration, XML and Web data management, and data integrity constraints. Dr. Mello is the head of UFSC Research Group on Databases (GBD-UFSC), having publications in national and international conferences and journals, and has served as external reviewer for national and international conferences and journals.