Self-Adaptive Data Quality Web Services

Tobias Vogel
Hasso-Plattner-Institut für Softwaresystemtechnik GmbH
Universität Potsdam
tobias.vogel@hpi.uni-potsdam.de

ABSTRACT

Data Quality Web Services are services that enhance the quality of data in the sense of making it fit for use. This paper concentrates on duplicate detection, which identifies multiple representations of real-world objects within large datasets where these representations are similar to a certain degree. Measures to estimate this similarity are one of the major research efforts in the community since many years.

However for Web Services, these heuristics differ in the amount of underlaying meta information, which is usually much poorer. The type of missing meta data is analyzed and classified in this paper. It also examines on different ways of making the duplicate-containing data available to the Data Quality Web Service.

1. DATA QUALITY

Data quality plays an important role for entrepreneurial success. However, many companies do not recognize the importance of data quality in their ERP or CRM systems, as recent studies show. Many different technical measures can be employed to increase data quality, e.g., data normalization, duplicate detection, and data fusion (Fig. 1). While the need to clean data is ubiquitous, particularly big enterprises spend money on dedicated data cleansing techniques and policies. To achieve this, they buy data cleansing frameworks and install and maintain them within their environments, not only spending money on license fees, but also paying the consultants or IT staff permanently, even if batch processing of their data only occurs once a month.

However, smaller enterprises share the same needs for good data quality, too. They need cleansing actions in the same frequency, while the amount of data might be smaller, but still too much for manual processing. E.g., for looking in a dataset 1,000 of, say, customer profiles, 500,000 comparisons are needed for finding duplicate (or near-duplicate) entries.

Due to the aforementioned difficulties and costs, measures ensuring data quality are often omitted. This can also be formulated as a need for ad-hoc, fair-priced, simple, low-configuration data cleansing. The *Software as a Service* paradigm promises to be a valid solution for this need since it employs services (most frequently in the shape of Web Services) as basic building blocks.

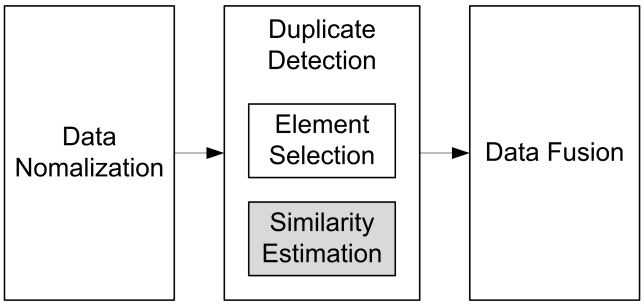

Figure 1: Data cleansing workflow

2. DUPLICATE DETECTION

Detection of duplicates is the process of identifying multiple representations of same real world objects. Traditional duplicate detection (also called *deduplication* or *record linkage*) employs well-established algorithms and heuristics, see Elmagarmid [2] and Winkler [11] for surveys.

Typically, those algorithms concentrate on (a) the selection of duplication candidates and/or on (b) a measure to estimate the similarity between two items. For (a) the goal is to find an elaborate selection of candidate pairs to avoid comparison of all ($O(n^2)$) pairs of elements, where the overwhelming number of comparisons will not be promising. For (b) the goal is to compare elements efficiently, i.e., with a good estimation of the actual similarity or in short time.

2.1 Web Services for Duplicate Detection

The comparison of two elements bases on data type and value of their attributes and additional information to identify these pairs as possible duplicates. However, sometimes the amount of available information is restricted or – as in Web Services – just not available: the schema might not be up-to-date, the field-mapping is unclear, privacy issues prevent full access to all the data, etc.

Thus, the question is how a good similarity measure can be created under the described conditions while still achiev-

ing appropriate results and while remaining as general as possible. Therefore, it has to be examined which information is essential for a duplicate detection process and which information therefore has to be inferred from the data or retrieved from other sources.

Web Service implementations of data cleansing – resp. duplication detection – methods (Data Quality Web Services) are invoked on-demand with exactly the information that is to be decided about, e.g., in case of only a small number of items that shall be tested for similarity in an ad-hoc manner. Further, they provide a clearly specified functionality while remaining as general as possible to ensure a broad number of possible service requesters. These properties turn Web Services into the ideal foundation for evaluating duplicate detection algorithms with the limitations described above.

2.2 Large Scale Duplicate Detection

Without the assumption of a Web Service to deduplicate only a small number of elements, further problems arise. The issue of how the data have to be made accessible to the services is covered by Faruquie et al. [3]. Furthermore, without this restriction, goal (a) becomes relevant in a Web Service scenario, too.

3. SIMILARITY MEASURES

Successful duplicate detection within unforeseeable data and structure requires some efforts before the actual detection of duplicates can take place. Different pieces of information may be missing, i.e., "semantics" of data, attribute names of data, mapping between fields, and attribute separators. All those terms are explained in the following. The lack of information can be seen as different levels of challenges for a good similarity measure. These levels are presented in the following.

3.1 Challenges for Similarity Measure Selection

Level 0 This represents the "traditional" scenario for similarity measures as depicted in Table 1. The record's attributes have proper names, the mapping is reliable (the last name of the first record has to be compared to the last name of the second record), and, of course, separators are available (represented by the table grid). Finally, the "semantics" are also clear due to the fact that the database managers are at hand and can provide this kind of information.

Therefore, specific measures such as a fine-grained heuristic for birthdates would detect that `1955|10|05` and `10.05.1955` mean the same date. Furthermore, it might be clear that the person's title is often guessed or omitted and thus, `Mr.` and an empty entry would match. The only requirement here is that these specialized similarity measures are available.

Level 1 If the data are provided without expert knowledge, e.g. in the XML-like shape illustrated in Figure 2, specialized similarity measures cannot be applied directly. First, attributes have to be classified. The result of this classification is some degree of certainty that a similarity measure is appropriate. To achieve this knowledge, the occurring values as well as the attribute names can be used. A particular challenge is to find out language-specific features, e.g., the order of month and day in a date field. Therefore, correlating data such as the domain of names or location information, should be considered, too. Note that the kind of representations is independent from the lack of expert knowledge.

Table 1: Homer Simpson (Relational Data)

Attribute	Record 1	Record 2
Title	Mr.	
First Name	Homer Jay	H. J.
Last Name	Simpson	Simpspn
Birthday	1955|10|05	10.05.1955

```
<address record="1">
 <title>Mr.</title>
 <firstname>Homer Jay</firstname>
 <lastname>Simpson</lastname>
 <birthday>1955|10|05</birthday>
</address>

<address record="2">
 <title/>
 <firstname>H. J.</firstname>
 <lastname>Simpspn</lastname>
 <birthday>10.05.1955</birthday>
</address>
```

Figure 2: XML

Level 2 Different records might have a different schema or no schema at all, e.g., because they come from different data sources or different owners. Figure 3 illustrates two records, where the first one is in XML format, the second one is in CSV format. It is obvious, that the schema is not clear. Not only are the entries in different order, also the occurrence of some attributes is different (e.g., the telephone number). Moreover, the granularity differs. The first record has separated title and name, the second one has separated family name from the rest.

It is commonly not clear, which attributes have to be compared with each other. The names of attributes might support the creation of a corresponding mapping, however those are not always available. Moreover, the telephone number of the second record might match the birthday of the first entry better than both birthdays match one another. Using meta information, such as thesauri or ontologies, help resolve synonym relations between terms, e.g., "surname" and "family name".

Level 3 The fields of a record might not be clearly distinguishable (c.f. Figure 4). E.g., the separator is not known or the record is represented as un- or semi-structured text. Therefore, the separator has to be found and entities have to be recognized, respectively. While the first separator (`;`) is rather common, the second one (`|`) is less frequent and might normally not be automatically recognized as an attribute separator.

```
<address record="1">
 <title>Mr.</title>
 <name>Homer Jay Simpson</name>
 <lastname>Simpson</lastname>
 <birthday>1955|10|05</birthday>
</address>

Simpson, Mr. Homer J, Springfield,
    10.05.1955, (019) 55 10 05
```

Figure 3: XML vs. CSV

```
Homer Jay;Simpson;742 Evergreen Terrace;Springfield

Mr.|Homer Jay|Simpson|742 Evergreen Terrace|Springfield

\charrsid87854 \par Homer Simpson}{\rtlch\fcs1
\af31507 \ltrch\fcs0 \lang1033\langfe1033
\langnp1033\insrsid6695672\charrsid87854
\par }{\rtlch\fcs1 \af31507 \ltrch\fcs0
\lang1033\langfe1033\langnp1033\insrsid87854
\charrsid87854 \par 742 Evergreen-Terrace \par }
{\rtlch\fcs1 \af31507 \ltrch\fcs0 \insrsid87854
Springfield \par USA \par \par }{\*\themedata 504b03
```

Figure 4: Different separators

The third example in this figure shows an RTF snippet kwhere these terms occur, too.

3.2 Similarity Challenges Tree

The challenges presented in Section 3.1 can be arranged in a tree as pictured in Figure 5. Each black node represents the decision for the existence of a specific piece of information, starting from the top (whether or not attribute separators are available) and reaching to the bottom (whether or not insights in the "semantics" of the attributes are present). If the corresponding information is available, the left subtree is investigated further, else the right one. Each pass leads to a leaf node that holds one of seven scenarios which have been elaborated on in the previous section. The best results for a similarity measure can be achieved if the left-most scenario is selected.

Read the tree in the following way: For example to infer a mapping between the attributes, the separators (resp. the entities and their attributes) have to be identified before. This is also the reason for the tree being very unbalanced. For example, it makes no sense to try to detect the data type if it is unclear, where one attribute ends and another one starts. However, there might be an overlap for this detection process. To find a valid mapping it can be necessary to check the similarity of different candidate pairs on an attribute value level. Recall the example in Figure 4. When creating a mapping between the two record's attributes, different matching candidate pairs have to be examined, here `<birthday>1955|10|05</birthday>` with both, `10.05.1955` and `(019) 55 10 05`.

4. DEALING WITH LARGE AMOUNT OF DATA

As described as goal (a) earlier, checking for duplicates in a large set of data poses additional challenges, which are discussed here.

4.1 Data Transfer

There are three principles of how data transfer between the customer's database and the processing unit (the Web Service) can be done. All have different requirements on network bandwidth, ease of use and requirements towards the customer's infrastructure.

Bulk File Transfer The relevant records can be exported from the customer's database into a flat file and then be transfered as-is to the Web Service. Technical restrictions such as the file being too large can be tackled by compressing the file or by splitting the file into smaller parts. The overall network usage is minimized here, first if compression is used and second, because each record is only transferred once. Moreover, the transfer is rather atomic from the point of view of the customer. There is no need to continuously being in charge of providing data.

ODBC Access The Web Service could be provided with a possibility to directly access the customer's database. In this case, Web Service (SOAP) requests are only needed for the control connection, the actual data is transferred directly. However, this is only possible, if the customer is able and willing to provide a way to access the database. Most probably, the network topology (firewalls, NAT) and security policies will complicate this access. The overhead is also higher in this approach, because the Web Service will either send many independent queries to the database or (most probably) will just issue a `SELECT * FROM relevant_records` query to retrieve the needed data. Then, the data transfer is not as condensed as the approach above.

Web Service The third possibility for the customer is to provide a (Web Service) API to his database. This will on the one hand reduce network topology problems, but on the other hand drastically increase bandwidth usage and CPU load, since much SOAP en- and decoding is to be under way. Furthermore, the customer has to provide an environment where this API is embedded. However, if the total amount of data is fairly small, the API already exists and the duplicate detection shall be very autonomous, this might also render an appropriate approach.

4.2 Candidate Selection

If the data is on the Web Service's side once, processing can begin. Assuming that the challenges described in Section 3 are successfully managed, duplicate detection can take place as usual. However in this scenario, many customers are to be served at the same time (with potentially also large amounts of data). Therefore, parallelized algorithms for searching for duplicates in relatively promising subsets of the whole dataset are needed.

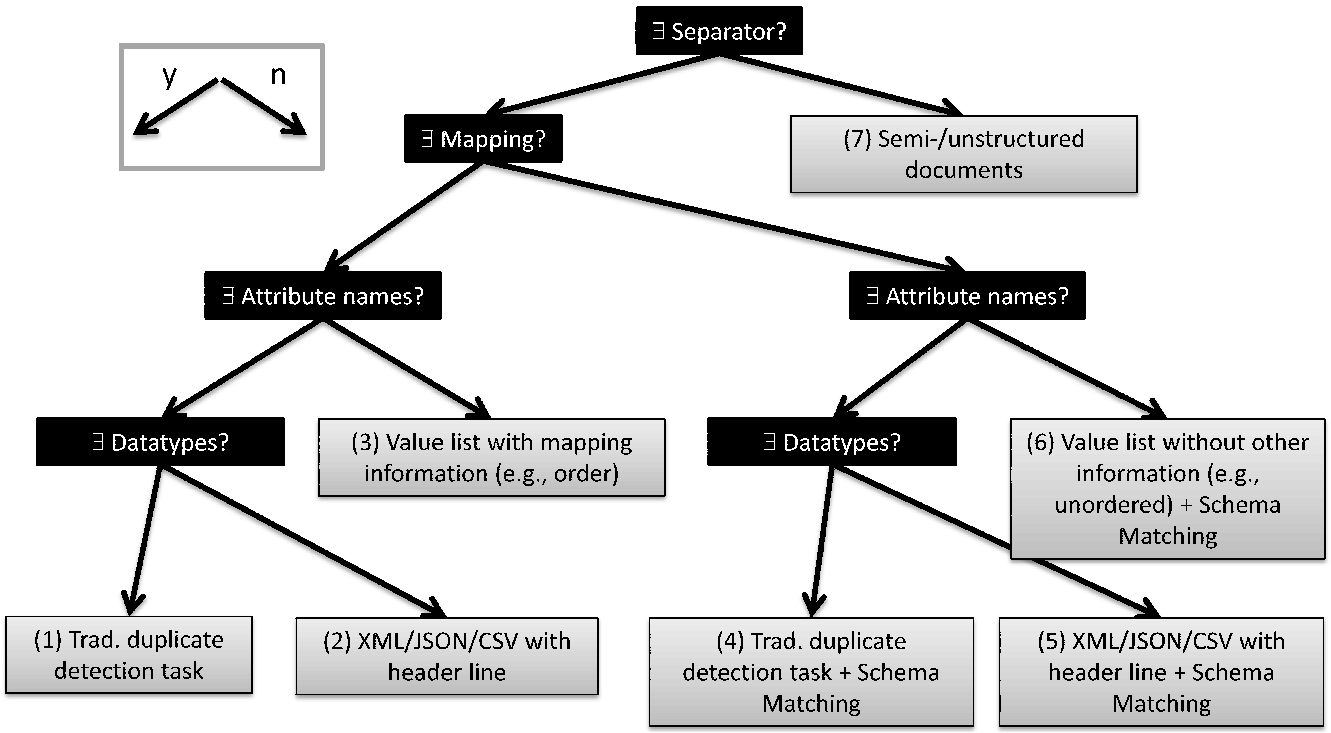

Figure 5: Tree of Different Classification Certainty Levels

In principle, two basic algorithms are widely used, *Sorted Neighborhood* and *Blocking*. In the Sorted Neighborhood approach [4], the whole dataset is sorted after specific sorting keys and then iterated over with a sliding window comparing all adjacent elements within this window. The other approach, Blocking, splits up the dataset into several smaller sets within all contained elements are compared pairwise.

Blocking is seen as the algorithm of choice [3, 8] when parallelizing algorithms. With this, different processing units can work independently from each other on different subsets of data, thus enabling a near-linear scale. However, selecting proper subsets is crucial since no inter-subset-duplicates can be found.

5. RELATED WORK

The different classes are already in focus of ongoing research, separately. For example, duplicate detection in XML structures [10] is examined by Weis et al.

Research in duplicate detection for unstructured or semi-structured texts is often applied in plagiarism detection systems as *MOSS* (Schleimer et al. [9]) or *YAP3* (Wise [12]).

Navarro [7], Winkler [11], and Elmagarmid et al. [2] give surveys on existing string comparison algorithms, based on edit distances as for example the Levenshtein Distance [6] and outline the specific usefulness for the corresponding values.

Faruquie et al. [3] examines challenges and benefits of bringing data cleansing (including duplicate detection) into the cloud. However, he does not provide any details on the proposed similarity measure or recommendations on the large scale data discussion points covered in Section 4.

There are a couple of approaches for parallelizing data cleansing, for example *Anthill* (Santos et al. [8]), *Febrl* (Christen et al. [1]), and *P-Swoosh* (Kawai et al. [5]).

There are a number of providers and registries for Data Quality Web Services. They provide services for data alignment (e.g., format telephone numbers properly), data completion (e.g., add postcodes to cities), or data verification (e.g., check ISBNs against book titles). However, there are no Web Services for duplicate detection.

6. SUMMARY

The process of *Data Cleansing* commonly comprises the normalization of data, the identification of duplicates within these data and the elimination of these duplicates by fusing or discarding duplicate elements. Employing services for this task is beneficial, because set-up and maintenance costs are low and they are only paid for on a by-use scheme.

Duplicate detection commonly faces two challenges: having a good, efficient similarity measure and selecting the "right" records to compare. In a Web Service application, the similarity measure is especially hard, since one or more of different (meta) data can be missing: "semantics", datatypes, mappings, and field separators. These information have to be inferred from the available data to decide for appropriate

comparison operators.

Another challenge is the selection of promising duplicate candidates which becomes even more tricky when it comes to parallelization efforts. Depending on this is also the decision on the way of how to bring the customer's data to the service. Currently, there is no best practice commonly agreed upon.

7. REFERENCES

[1] P. Christen, T. Churches, and M. Hegland. A parallel open source data linkage system, 2004.

[2] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge & Data Engineering*, 19, 2007.

[3] T. Faruquie, H. Prasad, V. Subramaniam, M. Mohania, G. Venkatachaliah, S. Kulkarni, and P. Basu. Data Cleansing as a Transient Service. In *Proceedings of the ICDE*, 2010.

[4] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. *SIGMOD Rec.*, 24(2):127–138, 1995.

[5] H. Kawai, H. Garcia-Molina, O. Benjelloun, D. Menestrina, E. Whang, and H. Gong. P-swoosh: Parallel algorithm for generic entity resolution. Technical Report 2006-19, Stanford InfoLab, September 2006.

[6] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Technical report, 1966.

[7] G. Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 1999.

[8] W. Santos, T. Teixeira, C. Machado, W. M. Jr., R. Ferreira, D. Guedes, and A. S. D. Silva. A Scalable Parallel Deduplication Algorithm. *Symposium on Computer Architecture and High Performance Computing*, 0:79–86, 2007.

[9] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *Special Interest group on Management of Data Conference*, 2003.

[10] M. Weis and F. Naumann. DogmatiX - Track Down Duplicates in XML. In *Special Interest group on Management of Data Conference*, 2005.

[11] W. E. Winkler. Overview of Record Linkage and Current Research Directions. Technical report, Bureau of the Census, 2006.

[12] M. J. Wise. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. In *Twenty-Seventh Special Interest Group on Computer Science Education Technical Symposium*, 1996.