An Analysis of Duplicate on Web Extracted Objects*

Stefano Ortona
Department of Computer Science, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD
stefano.ortona@cs.ox.ac.uk

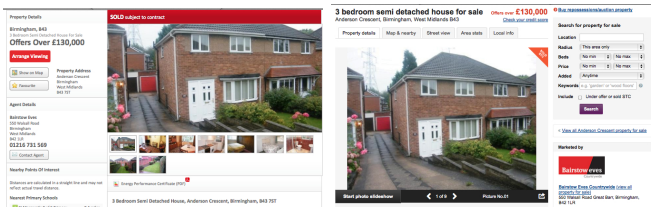
ABSTRACT

Today the web has become the largest available source of information. The automatic extraction of structured data from web is a challenging problem that has been widely investigated. However, after the extraction process, the problem of identifying duplicates among the extracted web records must be solved in order to present clean data to the final user. This problem, also known as record linkage or record matching, has been of central interest for the database community; however, only few works have addressed this problem in the web context. In this paper we present *web object matching*, the problem of identifying duplicates among records extracted from the web.

We will show that in the web scenario we need to face all the problems of a classic record linkage setting plus the uncertainty introduced by the web. Indeed the records are the output of an extraction system that, rather than conventional databases or APIs, introduces semantic errors that are not due to a problem in the source. Most of the previous approaches rely on the fact that the records to match contain the correct information and we can use such information to identify duplicates. In this work we overview an approach that performs a validation step before the actual identification of duplicates, in order to check whether the information of the record can be trusted or not. We present an approach that works without any human supervision or training data and that deals with the problem not only in a record-by-record fashion (as other approaches), but also in a source-by-source fashion which allows detecting and possibly correcting systematic errors for an entire source. The only human effort required is the creation of a little knowledge about the domain of interest through a set of ontology constraints and an entity extraction system.

*The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement DIADEM, no. 246858. This research has been supported by a grant from Amazon.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2579708.



(a) www.bairstoweves.co.uk (b) www.zoopla.co.uk

Figure 1: Two house offers from two different websites

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.5 [**Information Storage and Retrieval**]: On-line Information Services—*Web-based services*

General Terms

Experimentation, Algorithms

Keywords

record linkage; deduplication; matching; data cleaning; data extraction

1. INTRODUCTION

Today the web has become the largest database ever built in human history. Web users search more often for specific *objects* of interests such as people (e.g., Facebook), real estate (e.g., Rigthmove), or products (e.g., Amazon) rather than simple keywords to be found in web pages. Unfortunately, current search engines often fail to provide structured and precise information about specific web objects because they do not attempt to understand data as belonging to individual objects, due to the vast scale of the web. Recently search engines are trying to fill this gap (e.g., Google Knowledge Graph), however the results are still insufficient for the final user. Infact if we query Google with *''house 4 bedroom 2 bathroom London city centre"* the result is a list of websites that needs to be inspected by the user to find the desired result. The problem of automatically extracting complex objects from web is a big challenge and much work has tried to target this problem [8, 12, 7]. Such systems, given a set of websites as input, try to automatically create the programs that are able to extract the objects of interest from the input websites. The generated programs are often defined as *wrappers*. The creation of wrappers that correctly extract all the interesting data from the target websites is a very challenging problem and even if we are able to create such wrappers, after the extraction process new problems arise. Indeed same objects might

Price	Location	Postcode	Type	Status	#Bedroom	#Bathroom	Street	City	Town	Description
Weybridge - Guide Price £4,000,000	Weybridge - Guide Price £4,000,000	NULL	NULL	NULL	11	5	NULL	NULL	NULL	Weybridge

Table 1: Real estate record extracted from www.curchods.com

reside at different sources (each website is a source of information) and after the extraction process the output may contain several duplicates, that is several records that refer to the same real-world entity. Web object matching is the problem of identifying duplicates among a set of records (objects) extracted from web pages.

The problem of identifying duplicates among records coming from different databases is known as record linkage, record matching or de-duplication and it is one of the steps of the more general problem of data integration [22]. Web object matching adapts the setting of a classic record linkage problem to the web scenario, where the records come from web pages rather than databases. Objects that refer to the same real entity reside at different sources, where each source is a website that publishes the data according to its own schema. Such schemas are unknown to the user and also there is no guarantee that all the objects within the same website are published according to the same schema. Therefore same attributes may have different names in different sources, some attributes may be omitted in some sources because considered irrelevant and values for the same attributes coming from different sources may be in conflict with each other. These are the classic problems of a record linkage system and we can define them as *record-level* errors, due to the quality of the source irregularity. Note that such errors would occur even when the wrappers are manually created, as the source lacks the information or the wrapper generated does not work on all the pages of the website for a template change. However, when we deal with web databases, even greater challenges are encountered. First of all a website is not a normal database and is not forced to follow the same constraints. We might have the exact same record repeated twice in the same website or some attribute values may not follow the attribute type. Such problems can be easily avoided when dealing with normal databases. Most importantly, we need to keep in mind that the records to match are the result of an extraction process, and this process is not likely to be perfect. Therefore the records to match may contain erroneous or partially true values, that means we cannot fully trust the values of the attribute for the matching process. Most of the previous approaches for web object matching do not take into consideration this aspect (e.g., [27, 1, 21, 15, 18]), they rely on the fact that the wrappers are perfect and all the values (when available) can be considered as true values. We can define such errors as *wrapper-level* errors, where situations such as attribute misalignment, record misalignment and even template misalignment (where multiple record types are mixed because the wrapper does not understand they belong to different templates) are all due to the automatic generation of the wrappers that causes the wrappers to behave as not expected. In this paper, other than classic record-level errors, we want to consider the wrapper-level errors as well. Figure 1 shows two different offers from two different websites that refer to the same house. The website Zoopla is an *aggregator*, a website that collects offers from several real estate agencies and publish them on its website, while Bairstow Eves is a real estate agency. The two offers look similar, however we can notice that the two images are slightly different and the post-code is fully specified for (a), while (b) publishes only the first three digits of the post-code. This is a typical example of what we can expect when comparing records coming from different websites. Moreover, web object matching has a practical application that every e-commerce aggregator needs to solve: *product matching*. In product matching the records to match are product offers and the sources of information are the e-commerce websites where the offers come from. Indeed when we query the Bing shop catalogue for a camera [18], we would like to have each result representing a different product rather than having seven different descriptions of the same camera.

In this paper we present an extensive overview of challenges of identifying duplicates among records extracted from web (section 2) and we give a preliminary approach on how to deal with such problems. The work in this paper is currently in use for identifying duplicates in DIADEM [12], a completely automatic web data extraction system. Our approach relies on little knowledge of the domain of interest represented through a set of ontological constraints and on an entity extraction system, the annotator. We divide the identification of duplicates into three major steps: **(i)** *validation*, where the annotator is used to check whether the attribute values are correct and if they can be trusted during the following phases; **(ii)** *blocking*, where all the pair of records that violate one or more ontological constraints are put in different clusters; **(iii)** *scoring*, where we compute a score for all the pair of records that are in the same cluster (the score is calculated as the weighted average of similarity functions computed on all the attribute values) and we output as duplicates all those pairs of records that have a score above a threshold. All the steps are explained in details in section 3. Our approach wants to target primarily those errors caused by the wrappers (wrapper-level errors) and allows us to deal with the uncertainty introduced by the extraction process. Unlike most of the previous works, we present an approach that does not require any human supervision or training data. The only human effort needed is building the ontological constraints, our little knowledge about the domain.

The benefits of such system are: **(1)** we can avoid duplicates when showing the results of the extraction to the final user; **(2)** we can enrich the information about a single object by combining data residing at different sources, increasing the quality of the data; **(3)** we can give a unified representation of the object (global view) that can be easily queried by the final user; **(4)** we can use the matched objects (and the merged data from different objects) to understand whether there was some kind of errors during the extraction process; these errors can be used by the extraction process as feedback to improve the extraction strategy itself, to create a loop where the extraction and the matching step both help each other.

2. WEB OBJECT MATCHING

DEFINITION 1. *Let $\mathcal{D}$ be a set of target websites and let $\mathcal{W}$ be a set of wrappers, where $\forall d_i \in \mathcal{D}$ we have one (and only one) $w_i \in \mathcal{W}$. For each website d_i we have one wrapper w_i to extract records from the website. We define $\mathcal{S}$ as our global schema and $\mathcal{R}$ as the set of records extracted from all the websites in $\mathcal{D}$ using the wrappers in $\mathcal{W}$. All the records in $\mathcal{R}$ follow the schema $\mathcal{S}$. The goal of web object matching is to identify pairs of records in $\mathcal{R}$ that refer to the same real-world entity, that means $\forall r_i, r_j | r_i \in \mathcal{R}, r_j \in \mathcal{R}$*

web object matching outputs true if the pair r_i, r_j refer to the same real-world entity and outputs false otherwise.

Web object matching takes as input a set of records $\mathcal{R}$ and identifies the pair r_i,$r_j \in \mathcal{R}$ that refers to the same real-world entity, a problem also known as record linkage. The records in $\mathcal{R}$ are extracted from web pages via a website's wrapper. The focus here is not on how these records are extracted from web pages, we suppose to have a set of wrappers (one for each website of interest) that produce as output a set of records according to a pre-defined schema. Many previous approaches target the problem of record linkage for multiple databases, that is a classic record linkage problem, in this work we focus on the web scenario where each website is a potential database. In such settings we face all the problems of a classic record linkage plus all the new challenges due to the web databases. Below we give an overview of what the overall scenario looks like.

Global and Local Schema. The records in $\mathcal{R}$ follow all the same schema $\mathcal{S}$. $\mathcal{S}$ is manually defined along with the wrappers and aims to represent all the relevant aspects of the domain of interest. In DIADEM, where the main domain of interest is real estate offers, $\mathcal{S}$ contains attributes such as Price, Location, Number_of_bedroom, Number_of_bathroom. The websites, instead, follow their own schema, the *local schema*, because each of them provide data according to its own publishing model. Even though the global schema tries to generalize each single local schema into a more abstract one, there is no guarantee to have a direct correlation between the local and the global schema. Therefore some attributes may be omitted for some local schema because considered irrelevant (not all the real estate websites provide the number of bathrooms), and this means having many null values for the records that are extracted from those websites. In addition to this, since the wrapper is not perfect, a null value may be caused for an error of the wrapper which is not able to locate and extract the attribute. Table 1 shows an example of a real estate records extracted with DIADEM. As we can see half of the attributes have a null values. Many previous approaches do not consider the possibility of having null values (e.g. [27]), however this aspect is important when dealing with web records and it is clear that a null value does not help the duplicates identification.

Erroneous and Partial Attribute Values. The records in $\mathcal{R}$ are the output of several wrappers. Building a system which is able to create a wrapper for any website of interest is a very hard problem, therefore we need to allow the wrapper to make mistakes. The wrapper, other than missing some values to extract from the page, may also extract erroneous values for the attributes. Consider again the record extracted in table 1. In this example it is clear that the value for the attributes Price and Location are partially correct (the values contain the correct information as well as some noise) while the value for the attribute description is totally wrong (this value instead corresponds to the attribute Town). We call these errors the *uncertainty of the data* extracted by the wrapper. Note that also manually created wrappers can make mistakes, due to a problem in the source or an unpredictable behavior of the website, but in our situation the errors introduced by the wrapper increase (significantly) the uncertainty of the data. All the approaches that try to eliminate duplicates for web records rely on the fact that the wrappers are perfect, that means the extracted records contain correct values of the attributes. Some of them allow the presence of null values or errors due to the source of information, but none of them consider the errors that might be introduced during the extraction step. Unfortunatelly a wrapper is likely to be perfect only when it is manually created, when we want to deal with wrappers that are automatically generated (like in DIADEM) we must allow a certain degree of errors. The linkage strategy needs to validate the attribute values (in order to understand whether a price is a real price), it needs to parse the attribute values (to prune out the noise of the value and keep only the relevant information), it needs to understand whether a value must be assigned to a different attribute (like the value of the attribute Description in table 1). All these validation steps must be done before the actual de-duplication phase and even after this pre-process we still need to allow some form of uncertainty (since we cannot be sure that the validation steps work perfectly). One may argue that these wrapper-level errors could be addressed also during the extraction process itself, however by adressing them in a post-process stage we can take advantage of having data (and redundancy) from different websites and therefore more knowledge.

Source of Duplication. In the classic record linkage problem we have two (or more) databases and these databases might contain duplicates. A pair of duplicates consists of two records that belong to two different databases and refer to the same real-world entity. When we consider records extracted from web pages the situation is more complicated and we have multiple reasons of duplication. The majority of the duplicates is due to the same reason of classic linkage setting: we have two or more websites that publish the same record (according to their own publishing model that is unknown to the user) and when we extract records from both websites we find the duplicates (see figure 1). However we have other factors that cause duplication. First of all the websites are not forced to respect any kind of constraints, and it might happen the same website publishes the same record several times. We have seen cases where the same record has been published on the same website with exactly the same attributes and values (this could be easily avoided in a traditional database using uniquess constraints) and even cases where two records refer to the same entity but their representations were different (on the same website). The first two records in table 2 represent two offers from www.kempandkemp.co.uk that refer to the same house. At first sight the offers may look identical, but with a deeper look we can notice that the attributes Location and Type are slightly different and Number_of_bedroom is null for one of the offers. Eventually, again the wrapper may introduce more errors, by adding more duplicates. In fact to visit an entire website and extract all the records from the website is very challenging (the wrapper needs to simulate the human behavior to navigate through the website pages) and it is likely that the wrapper makes some mistakes, like visiting the same page twice. Therefore the wrapper itself might introduce more duplicates by extracting the same record different times. One may argue that in this case the extracted records are exactly the same (and therefore easy to de-duplicate), however since the behavior of the wrapper (and of the website) is unpredictable the same record might be extracted with different values in two different times. To complicate even more the scenario, websites may change as they are being visited, and this could add more errors (as the wrappers are unable to understand the change of the content).

We have seen that web object matching needs to face all the problems of a classic record linkage setting and also new challenges due to the web uncertainty. Table 2 shows an example of real estate offers extracted with DIADEM that refer to the same house. The first two records come from the same website (Kemp and Kemp) while the last record has been extracted from Zoopla. As we can see we have null values, erroneous values, duplication intra website e inter website.

3. APPROACH

In this paper we present a dependency driven data cleaning system based on little knowledge on the domain of interest. Our framework takes as input some domain-specific knowledge and it uses

Price	Location	Postcode	Type	Status	#Bedroom	#Bathroom	Street	City	Town	Description
£1,350,000	41 Blandford Ave	NULL	41 Blandford Ave	Under Offer	5	3	NULL	NULL	NULL	5 bedrooms 3 bathrooms 3 reception rooms
£1,350,000	41a Blandford Ave	NULL	41a Blandford Ave	Under Offer	NULL	3	NULL	NULL	NULL	5 bedrooms 3 bathrooms 3 reception rooms
£1,350,000	Blandford Avenue, Oxford OX2	Blandford Avenue, Oxford OX2	5 bedroom detached new house for sale	for sale	NULL	NULL	Blandford Avenue, Oxford OX2	Blandford Avenue, Oxford OX2	NULL	Added on 20th Aug 2013

Table 2: Real estate records extracted from www.kempandkemp.co.uk and www.zoopla.co.uk that refer to the same real-world house

that knowledge to identify record duplicates. The main idea is that if we focus on just one domain, we can observe some repeating patterns that are domain-specific. By exploiting those patterns we are able to **(i)** parse and clean the erroneous and partially true attributes; **(ii)** identifying duplicates even when the information available is not complete. Our knowledge of the domain is made of an entity extraction system, the *Annotator*, and the *Ontology*, that contains repeated patterns that we can observe for each website of our domain of interest. In this paper our case study is the domain of real estate, where the records are real estate offers (both selling and renting offers) that are extracted from real estate websites.

The Annotator. The annotator is an entity extraction system that takes as input a free text and identifies the entities of interest. In the real estate domain such entities are, among the others, Location, Price, Number_of_bedroom. Our annotator is a combination between ROSeAnn [3], a domain independent entity extraction system, and a manually-created annotator (with Gate [10]). ROSeAnn is responsible of identifying those general entities such as Location, City, Person, it is not domain dependent and it can be used for any domain of interest; the domain-specific annotator identifies those entities that are domain-specific, in our case entities such as Price, Number_of_bathroom, Type_of_house. Note that the only domain-specific part for the annotator is the one created through Gate. This is essential to understand those specific concepts that can be found only in one domain, since all of the available entity extraction systems are able to identify only general concepts (see [3] for a general overview of entity extraction systems).

The Ontology. The Ontology contains those constraints that are domain-specific and that can be used to prune out some duplicate candidates. An example of constraint in the real estate domain is the type of contract agreed between the agency and the owner of the house. The contract can be exclusive (the selling/renting of the house is committed to one and only one agency) or it can be multilateral (several agencies try to sell/rent the house). Obviously we do not expect to find web-offers of the same house in more than one agency website if the contract of that house is exclusive, while we might find several offers from several websites of the same house if the contract is multilateral (note that we still may have duplicates for houses with exclusive contract because of the aggregator websites). Another example of constraint is that two offers cannot refer to the same house if one of them is a renting offer and the other one is a selling offer. Now it should be more clear how the ontology constraints might be helpful for identifying duplicates, especially for identifying those that are not duplicates. The Ontology also specifies those attributes of the records that we define *dominant*. A dominant attribute is an attribute that almost uniquely identifies the record and the use of the dominant attribute is explained below. In real estate Location is a dominant attribute while Type_of_house is not. Other than dominant or not, the ontology also specifies if an attribute is *atomic* or *compound*. An atomic attribute is an attribute that is not made of other attributes, like City or Number_of_bedroom. In real estate Location is an example of compound attribute, since it is made of several atomic attributes like Street_ address, Post-code, City etc.

The creation of the annotator and of the ontology are the only parts that requires human effort, however they need to be done just once for each domain of interest and once they are ready, they can be used for all the websites of the input domain (for DIADEM the creation of the annotator and the ontology required one week of one-person work). The main idea is that with an initial little effort we can automatize the process for all the websites of the whole domain.

A three-step algorithm. Our algorithm divides the duplicate detection phase in three major steps: *Validation*, *Blocking* and *Scoring*.

(1) In the validation step the goal is to analyze each attribute of the records in order to discover erroneous values, partially complete values or misplaced values. We run the annotator on each single attribute and on the concatenation of all the attributes in order to adjust the following situation: **(i)** an attribute value is wrong when we run the annotator on the value and the annotator does not return the type of the attribute at all; **(ii)** an attribute value is partially correct if we run the annotator on the value and the annotator returns the type of the attribute only for a substring of the value; **(iii)** an attribute value is correct if we run the annotator on the value and the annotator returns the attribute type for the entire value; **(iv)** an attribute value is misplaced when we run the annotator and the annotator returns a type that corresponds to another attribute type; **(v)** an attribute value is partially misplaced when we run the annotator on the concatenation of all the attribute values and we find out that the annotator returns an attribute type that is scattered across different attribute values. By using the annotator we are able to correct and check the attribute values such that they can be useful during the following phases. Also, by annotating each compound attribute, we try to extract the atomic attributes out of it (e.g., we could extract the attribute Post-code from the attribute Location). The validation phase has the goal to check the validity of each attribute but also tries to enrich the attribute values by inspecting each at-

tribute values. This step wants to reduce the impact of all the errors that are introduced by the wrappers, as we have seen in section 2. Moreover, other than a record-by-record comparison where we try to find the errors for a single record, in this step we also look at the repeated errors we may find for the same source (those errors that are exactly the same for many records of the same source). This may be caused by the source itself or by a repeated error of the wrapper. If this is the case, we come up with potential fixes that can be applied to the entire source at once. We need to keep in mind that even after this step the record's values may still contain errors, since the annotator is not able to correct some situations (if a value is wrong the annotator will not replace it with the correct value) and since the annotator itself may be wrong (the entity extraction systems are not perfect and may produce noisy annotations [8]). This is the main reason why we have introduced the domain's knowledge and the knowledge is exploited in the following steps to overcome the remaining errors.

(2) The blocking step identifies for each record all the records that cannot be a duplicate by using the ontology. For each pair of record r_i, r_j we check the ontology constraints and if one of them is violated, the two records are not put in the same cluster. An example of violation of constraint is when we have validated a dominant attribute for two records and the two records have that dominant attribute different (in real estate if we are sure that two houses have different location we are also sure that the two houses are not the same). After the blocking step all the records are divided into clusters such that each record belongs to one and only one cluster and there are not two records in the same cluster that violate one of the ontology constraints.

(3) In the last step we analyze all the clusters. For each cluster, we compute a score for each pair of records in the cluster and we output as duplicates all the records that have a score above a pre-defined threshold. The ontology, other than constraints, also defines what kind of similarity-string measure needs to be used for each attribute value (examples of similarity distance measures are Cosine similarity, TFIDF similarity, Levenstein similarity; see [6] for a general overview) along with the weights (the importance) for each attribute. The score is computed as a simple weighted average of the string similarity scoring computed for each attribute. Eventually all the pairs of records within the same cluster that have a score above the threshold are duplicates, while all the other pairs of records are not duplicates.

The ontology and annotator play a crucial role in the overall matching step but the benefits of using them are enormous. First of all we can reduce the uncertainty introduced by the extraction process that cannot be underestimate as pointed out in section 2. Secondly our approach does not require any kind of supervision or training data. Most of the record linkage algorithms try to solve the problem by learning some kind of similarity functions between records. In our setting the collection of training data is very hard due to the high variety of the data on the web and also (and most importantly) because of the unpredictable behavior of the wrappers. In our approach we require little knowledge about the domain of interest that allows us to face all the problems that are introduced by web records. One may argue that making a pairwise comparison among all the records is not feasible, especially during the blocking step when we need to check the ontology constraints for each pair of record. We consider this problem as orthogonal and we could apply any of the blocking techniques that have been proposed in literature (see [5] for a survey) to achieve scalability. Such blocking strategies reduce the number of pairwise comparison by grouping the records in different blocks, according to a pre-defined strategy.

Boost the Extraction. Eventually the information discovered by the matching step can be used as feedback to improve the overall extraction system, especially during the validation step. In fact during the validation step we may find out that the value extracted for the attribute post-code is not a real post-code, or the value extracted for the attribute town is actually a price. All such information can be used by the extraction process to better locate the value to extract from the web pages. Also, other than a single record fixes, we can identify source-level fixing, where we could inform the wrappers to fix an error for the entire source (few record pairs may be enough to identify systematically errors and to generalise for the entire source). The extraction and the de-duplication of the data have been always considered as two separated steps, however making the two of them cooperating in a continuos loop could improve the both of them, similar to what has been done in [2], where the redundat (and overlapping) data across different web sources is used during the extraction process to improve the quality of the wrappers by chosing those X-Path rules that best locate the redundant data on the new web pages to analyze.

4. RELATED WORK

Web object matching implies several techniques from data integration, data cleaning and web data extraction. Below we review some of the most relevant works.

Record Linkage. Record linkage is a problem that has been widely studied in the context of data integration [22] (see [20] and [9] for a survey). It addresses the task to identify records that represent the same real-world entity, where the records come from different databases with (possible) different representations (local views). Most of the techniques adopt a framework that uses three consecutive steps: **(i)** *blocking*, to create a set of small blocks, where each block contains similar records; **(ii)** *pairwise matching*, to compare all the records within the same block; **(iii)** *clustering*, to group the records that refer to the same entity within the same block. The typical setting of record linkage problem is to define a similarity function between records that returns a value of similarity, then use a threshold to understand whether two records refer to the same entity or not. The similarity function can be manually defined [17], but it requires domain experts, or it can be learned via a supervised algorithm [4]. In [29] a supervised approach is used to learn what is the best similarity function to use given a set of rules, however the idea relies on the assumption to have all the records coming from the same database. The problem of automatically generating training data for a binary classifier is addressed in [27]. The main idea is to create an inital set of positive (negative) examples by selecting the records having all the attributes identical (different), and then refine the training set with more examples. Even though the method does not require any human supervision, there is no guarantee about the presence of totally identical (differnet) records. Some techniques try to avoid the use of supervised algorithms by leveraging on temporal information [23, 1], using the crowd instead of human supervision [28] or by exploiting the relationship between different entity types in order to improve the linkage among entities with the same type [30]. Eventually [19] is a deduplication framework built on hadoop that can be easily integrate to have better performance and good scalability.

Web Record Linkage. Recently the focus of record linkage has moved from the classic database to the web scenario, where each website becomes a source of information and we want to link records from different websites. In the web scenario, where the data to match are often generated on-the-fly, the supervised approaches become inappropriate. Moreover the records to match are the output of a data extraction system; such systems have a behavior which

is hard to predict (due to the high variability of the web sources) and a learned method that uses training data collected today may fail on the data extracted tomorrow. In [27] the training data is automatically generated by the combination of two different classifiers that cooperate together, where the first classifier provides the training examples for the second classifier. The approach does not allow the records to have null-value attributes, but for data extracted from the web this aspect must be taken into account since the extraction system may fail to extract the attributes or the attributes might simply not be there to be extracted. The most practical use of record linkage in the web is matching product offers, where the entities are products and the extracted records are product offers. Both [21] and [15] try to identify the attributes of the offer that uniquely identify the product. The former extracts the product code with manually created rules while the latter enriches the title of the offer with a search engine in order to identify the keywords in the offer. Kannan et. al. [18] look at the optimal parsing of the offer w.r.t. a database of all products in order to find the parsing that maximizes the similarity score between duplicate records. The approach requires a database of all the entities to compare. All these approaches try to adapt the linkage techniques to the web setting, however none of them consider the critical aspect of the uncertainty of the data. As we have seen in section 2 the output of a web data extraction system is rarely perfect, some attributes are null, some attributes have an incomplete value, some attributes have a wrong value. A linkage system that wants to deal with such kind of data cannot avoid this peculiarity.

Data Cleaning. The goal of a data cleaning system is to make the database records consistent w.r.t. a set of given constraints [25]. We have seen that the records we want to link are the output of an extraction system that could introduce some errors in the attribute values, applying a cleaning strategy to these wrong values may help the overall linkage system. Fan et. al. [11] inspect the interaction between record linkage and data repair and try to unify these two steps. [24] suggests a knowledge-based framework for data cleaning to detect and eliminate duplicates, while [13, 14, 26] are interactive data cleaning systems that can be tuned directly by the end user. All these techniques can be used in a pre-process step to clean the wrong or incomplete values of the record, such that we can reduce the effect of the uncertainty of the data.

Web Data Extraction. Web object matching tries to identify duplicates from a set of web records extracted by a web extraction system, such systems are often defined as website *wrapper*. In the last few years the problem of web data extraction has become very important and many approaches have been proposed to solve it. Some approaches leverage on the redundancy of the data [16] and on the overlapping of the data [2] among different web sources in order to generalize a manually created wrapper or to prune away those wrappers that are not able to extract all the data from the website. Systems such as [12] or [8] try to understand the structure of the website in order to automatically extract complex entities and their properties by leveraging on the repeated structure of the web pages (within the same website) or by exploiting background knowledge of a specific domain. Due to the high variety of the web sources, all these systems are not able to produce perfect wrappers and the extracted records will always contain erroneous or incomplete values. The record linkage system must take into account such aspects and by identifying duplicates among several records we are also able to correct the wrong values and to enrich the information about a single record. Eventually the information about erroneous values can be given as feedback to the wrapper itself to improve the overall extraction process.

5. REFERENCES

[1] R. Agrawal and S. Ieong. Aggregating web offers to determine product prices. In *Proc. of KDD*, 2012.

[2] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti. Extraction and integration of partially overlapping web sources. *PVLDB*, 6(10), 2013.

[3] L. Chen, S. Ortona, G. Orsi, and M. Benedikt. Aggregating semantic annotators. *PVLDB*, 6(13):1486–1497, 2013.

[4] P. Christen. Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proc. of KDD*, 2008.

[5] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *KDE*, 24(9), 2012.

[6] W. W. Cohen, P. D. Ravikumar, S. E. Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78, 2003.

[7] V. Crescenzi, P. Merialdo, and D. Qiu. A framework for learning web wrappers from the crowd. In *Proc. of WWW*, pages 261–272, 2013.

[8] N. Dalvi, R. Kumar, and M. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4), 2011.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *TKDE*, 19(1), 2007.

[10] H. C. et. al. *Text Processing with GATE (Version 6)*. U. Sheffield Dept. of CS, 2011.

[11] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. In *Proc. of SIGMOD*, 2011.

[12] T. Furche and et. al. Diadem: domain-centric, intelligent, automated data extraction methodology. In *Proc. of WWW*, 2012.

[13] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: an extensible data cleaning tool. *SIGMOD*, 29(2), 2000.

[14] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The llunatic data-cleaning framework. *PVLDB*, 6(9), 2013.

[15] V. Gopalakrishnan, S. P. Iyengar, A. Madaan, R. Rastogi, and S. Sengamedu. Matching product titles using web-based enrichment. In *Proc. of CIKM*, 2012.

[16] P. Gulhane, R. Rastogi, S. H. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. *PVLDB*, 3(1-2), 2010.

[17] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406), 1989.

[18] A. Kannan, I. E. Givoni, R. Agrawal, and A. Fuxman. Matching unstructured product offers to structured product specifications. In *Proc. of KDD*, 2011.

[19] L. Kolb, A. Thor, and E. Rahm. Dedoop: efficient deduplication with hadoop. *PVLDB*, 5(12):1878–1881, 2012.

[20] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1-2), 2010.

[21] H. Köpcke, A. Thor, S. Thomas, and E. Rahm. Tailoring entity resolution for matching product offers. In *Proc. of EDBT*, 2012.

[22] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS*, 2002.

[23] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *PVLDB*, 4(11), 2011.

[24] W. Lup Low, M. Li Lee, and T. Wang Ling. A knowledge-based approach for duplicate elimination in data cleaning. *IS*, 26(8), 2001.

[25] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4), 2000.

[26] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, 2001.

[27] W. Su, J. Wang, and F. H. Lochovsky. Record matching over query results from multiple web databases. *TKDE*, 22(4), 2010.

[28] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11), 2012.

[29] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: how similar is similar. *PVLDB*, 4(10), 2011.

[30] S. E. Whang and H. Garcia-Molina. Joint entity resolution. In *ICDE*, 2012.