Big Data Integration for Product Specifications

Luciano Barbosa[1], Valter Crescenzi[2], Xin Luna Dong[3], Paolo Merialdo[2], Federico Piai[2],
Disheng Qiu[4], Yanyan Shen[5], Divesh Srivastava[6]
[1] Universidade Federal de Pernambuco luciano@cin.ufpe.br
[2] Roma Tre University {name.surname}@uniroma3.it
[3] Amazon lunadong@amazon.com
[4] Wanderio disheng@wanderio.com
[5] Shanghai Jiao Tong University shenyy@sjtu.edu.cn
[6] AT&T Labs – Research divesh@research.att.com

Abstract

The product domain contains valuable data for many important applications. Given the large and increasing number of sources that provide data about product specifications and the velocity as well as the variety with which such data are available, this domain represents a challenging scenario for developing and evaluating big data integration solutions. In this paper, we present the results of our efforts towards big data integration for product specifications. We present a pipeline that decomposes the problem into different tasks from source and data discovery, to extraction, data linkage, schema alignment and data fusion. Although we present the pipeline as a sequence of tasks, different configurations can be defined depending on the application goals.

1 Introduction

The product domain represents a challenging arena for studying big data solutions that aim at performing data integration at the web scale. An impressive number of product specification pages are available from thousands of websites, each page associated with a product, providing a detailed product description that typically includes technical and physical features, price, and customers' reviews. Integrating the data offered by these pages to create a comprehensive, unified description of each product represents a foundational step to enable valuable applications, such as, question answering, price comparison and data driven market analysis.

In this paper we describe an overview of our ongoing research activities to address the issue of integrating data extracted from product specification pages. We present some results that we have already achieved, the lessons that we have learned from our experience, open problems and future directions.

1.1 End-to-End Data Integration Pipeline

We refer to an end-to-end data integration pipeline for product specification pages that includes several tasks, as depicted in Figure 1. Every task adopts methods and techniques from databases, information retrieval and

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

machine learning. Every task produces output to feed the successive task in the pipeline, but intermediate results could find other compelling application scenarios as well. To this end, we advocate that the pipeline must include an empirical evaluation benchmark for every task.

In our vision, the information need is expressed by an input set of sample pages. We observe that products are typically organized in *categories*, and hence we expect that the sample pages refer to products from the categories of interest. Our approach is inspired by the Open Information Extraction [6] paradigm: the schema for the target data is not specified in advance, and the categories of the target products do not refer to a predefined product taxonomy, but they are rather inferred from data in the product pages of the input sample and in the product pages that are gathered from the Web along a progressive and iterative process.

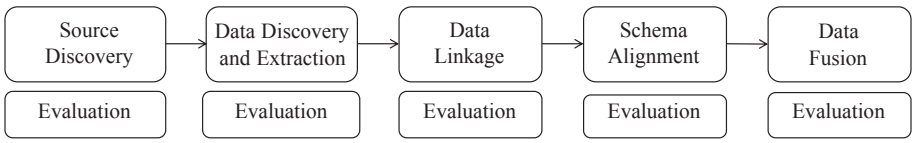

Figure 1: Our end-to-end big data integration pipeline for product specifications.

Source discovery aims at efficiently finding and crawling product websites in order to gather pages that refer to the products of interest. One might believe that discovering product websites is a minor task, as a relatively small number of *head sources* can offer enough data for most of the products. For example, amazon.com already provides data about an impressive number of products. However, valuable information is actually published by an enormous number of *tail sources* [3, 4], i.e., sources that each provide a small number of product entities. These tail sources are important because they improve coverage. They can often offer *tail entities*, i.e., products that are present in a small number of sources, as well as *tail attributes*, i.e., product properties that are present in a small number of entities. Also, tail sources ofter refer to *tail categories*, i.e., small niche categories of products. Finally, *tail sources* contribute to information diversity, as they provide values that depend on the local source, such as, product reviews and price. Source discovery also deals with efficiently crawling the discovered websites towards pages containing products of the categories of interest, without visiting unproductive regions.

Data discovery and extraction has the objective of processing the pages harvested in the previous task in order to locate and extract product attribute names and their values. As we mentioned above, we do not rely on a predefined schema, but rather extract attributes bottom-up, with the goal of discovering not just *head attributes*, but also *tail attributes* that cannot always be described in advance.

Data linkage seeks to cluster pages from different sources that refer to the same products. It is worth observing that in the traditional data integration pipeline, schema alignment is performed before record linkage [5]. Unfortunately, with a very large number of sources, such a traditional approach becomes infeasible because of the huge variety and heterogeneity among attributes. As we shall discuss later, we propose to perform data linkage before schema alignment as we can take advantage of the opportunity that products are named entities, and hence a product specification page usually publishes the product identifier.

Schema alignment addresses the challenge of semantic ambiguity and aims to reconcile the attributes offered by different sources, that is, to understand which attributes have the same meaning and which ones do not, as well as identify value transformations to normalize different representations of the same attribute values. Since we do not rely on a global schema given in advance, correspondences among attributes are established bottom-up leveraging the results of the previous data extraction and data linkage phases.

Data fusion tackles the issue of reconciling conflicting values that may occur for attributes from different sources. Data fusion aims at evaluating the trustworthiness of data, deciding the true value for each data item,

and the accuracy of the sources. To address these challenges, data fusion techniques rely on data redundancy, which further motivates the need to process many sources.

For simplicity of presentation, we have described our approach as a linear pipeline, where tasks are performed in sequence and independent of one another. However, there might be feedback loops between the tasks, as intermediate results can indeed influence the performance and the behavior of the preceding tasks and of the end to end solution.

1.2 Challenges and Opportunities

The web scale raises intriguing *challenges* for all the tasks of our pipeline due to its volume, variety, velocity, and veracity [5].

- The *volume* of data refers not only to the large number of products but, more importantly, to the number of sources. As we have discussed, to achieve coverage and diversity, we need to process a very large number of sources across the entire web, not just a small number of pre-selected sources.
- The *variety* of data is directly influenced by the number of sources and arises at many different levels, affecting every task of our pipeline. At the product category level, websites, especially the head ones, organize products according to such a vast plethora of categorization strategies that makes it very difficult, if not impossible, to reconcile them into a unified taxonomy. At the product description level, heterogeneities are related to attributes and values, which are published according to different granularities (e.g., physical dimensions in one field vs three separate fields for width, length, height), formats (e.g., centimeters vs inches) and representations (e.g., the color of a product as a feature vs distinct products for different colors).
- The *Velocity* of data involves the rate of appearance and disappearance of pages in sources as well as the rate of appearance and disappearance of web sources. Also, while some attributes (such as technical and physical features) are quite stable over time, the contents of the individual pages can change daily, for example for prices and reviews.
- The *Veracity* of data deals with honest mistakes that can occur in web pages, but also with deceits, that is, deliberate attempts to confuse or cheat (e.g., providing imprecise or erroneous product characteristics).

Our approach to address these challenges aims at taking advantage of the *opportunity* that products are named entities, and hence a product specification page usually publishes the product identifier. Web sources that deliver product specification pages publish product identifiers mainly for economic reasons: websites need to expose the product identifiers to let them be indexed by shopping agents and available to customers who search products for comparing prices or consulting specifications. Large e-commerce marketplaces strongly encourage sellers and retailers to publish product identifiers,[1] as they improve efficiency both for the internal management of data and for the exchange of data with search engines like Google and Bing.

The presence of identifiers allows us to drive the pipeline from source discovery to data integration by leveraging the opportunity of *redundancy of information at the global level*, and the *homogeneity of information at the local level*.

- At the global level, we observe that head (popular) products are present in several head (large) sources as well as in many tail (small) sources. Therefore, we expect that identifiers of head products are spread across many sources. Further, many head products in a category will often co-occur in multiple sources.

[1] eBay, Amazon, Google Shop explicitly require sellers to publish the id for many product categories. For example, see eBay's rules: `http://for-business.ebay.com/product-identifiers-what-they-are-and-why-they-are-important`.

- At the local level, we observe that the structure and the semantics of information, within each source, tend to be regular. Hence, we expect the product specification presented in a given page is published according to the same structure for every page in the same source.

Redundancy as a Friend Figure 2 illustrates the key intuitions underlying our approach [11] from source discovery to data integration to meet the goal of *effectively* and *efficiently* dealing with head and tail sources, hence including all pages of head and tail entities. Starting from known head entities in head sources, we take advantage of homogeneity of information at the local level to extract product specifications and identifiers for tail entities in head sources (even head sources offer many tail entities). Then, we exploit the presence of head entities across sources: searching head identifiers, we discover tail sources (even tail sources offer a few head entities). Again, we exploit homogeneity of information at the local level to extract identifiers and specifications for tail entities in tail sources.

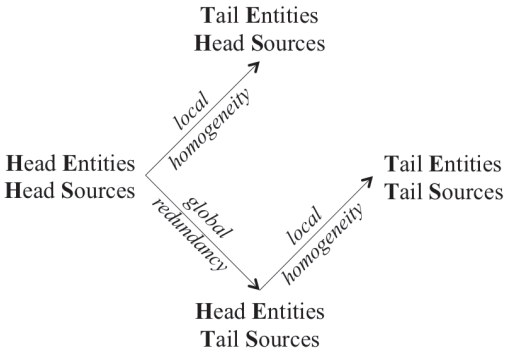

Figure 2: Our approach.

2 Source Discovery

We consider a focused crawler, Dexter [12], to discover and crawl product websites offering product pages for the input categories. The crawler iterates over three phases: *source finding*, *source crawling*, *identifier extraction*.

2.1 Source Finding

Our target websites are sparsely distributed on the web. To efficiently find them without visiting unproductive regions on the web, we consider two different strategies, *search* and *backlink*, whose results can be suitably combined. *Search* consists of querying a search engine with a set of seed identifiers of products; we expect that the search engine results contain pages from websites that publish information about these products. *Backlink* exploits services that provide inbound links, i.e., pages containing the set of links pointing to a given page; we rely on these services to find hubs, i.e., pages aggregating links to product websites.

These strategies allow us to discover many sources without penalizing recall, with Search focused more on head sources, Backlink including also tail sources. However, they often return also many non-relevant sites. To improve precision, we filter out results that are unlikely to represent a product website.

Search takes advantage of the redundancy of product information: searching the identifiers of known products on a search engine, we expect to find pages of the same products in many different sites. Identifiers to trigger the search engine are extracted from the input set of pages by means of suitable wrappers, or leveraging microdata annotations, such as schema.org, in case these are used in the pages.

Observe that the search engine can also return pages from websites that are not useful for our goals, such as pages from a web forum or pages from news websites. To efficiently select relevant websites, we search for multiple products and rank websites based on the number of pages from the same website that are present in the results. The rationale is that it is unlikely that a website that does not contain product specification pages appears frequently on the results of multiple queries with different product identifiers. Based on the ranking, we select the top-k websites.

It is worth observing that this strategy can penalize the recall of tail sources whose pages do not appear with a sufficient number of occurrences in the search engine results. This limitation can be further exacerbated by costs and limitations of the search engine APIs, which usually impose limits on the number of results per query, and on the number of queries in an interval of time. This is a significant issue if the goal is to collect an unbounded number of product websites.

Backlink aims to deal with the above limitations by adopting an alternate solution that is not dependent on the search engine restrictions. The idea is to find hubs, that is, pages that list links to many product websites. To this end we rely on online services that return the inbound links to a given input page.[2] Our approach is to search for pages that contain pages containing inbound links to the websites returned by Search. We then consider the domain of the links contained in these pages as candidate product websites. Similarly as for search engine results, backlinks can also lead to non-relevant hubs, subsequently leading to non-relevant websites. For example, sometimes they return generic hubs that point to multiple websites like popular websites in a country.

To select the most promising websites without penalizing the recall, we adopt an iterative approach to compute a ranking of the websites contained in the hubs returned by a seed set of product websites. As non-relevant hubs are less likely to point to many relevant websites, we score hubs based on the number of relevant websites they point to, and similarly we score the relevance of websites, based on the number of hubs that point to them. Based on this intuition, we rank websites and hubs and select the top-k websites.

Filtering Results The collection of websites discovered by Search and Backlink can contain many spurious results. In order to select our target websites, we adopt a simple machine learning approach, training a classifier to recognize if a website is a product website. The features that we have considered include all the anchor texts of the links in the home page.

2.2 Source Crawling

Source discovery also deals with crawling the discovered websites towards the product pages of the target categories. Also in this case, efficiency is a challenging objective, as websites often contain a huge number of pages, and only a small fraction may represent pages of interest.

We adopt a crawling strategy inspired by [8], which focused on web forums. Our approach builds on the assumption that, similar to forum websites, product websites have an internal structure consisting of one or more *entry pages* to the product content for a given category, followed by possibly paginated *index pages* that offer links to product pages. Therefore, our approach first looks for the entry pages related to the target category, then seeks for the index pages.

To discover the entry pages for a given category, the crawler relies on a classifier trained to predict links that are likely to point to entry pages. The classifier, which is trained with the links to entry pages of the target category in the websites of the input set of sample pages, analyses the anchors of links that are present in the home page and in target product pages (those returned by the search engine in the source discovery phase).

A similar approach, based on machine learning, has been adopted also to detect index pages. In this case, the classifier is trained to predict links that lead to product pages. A page is considered an index page if it contains a collection of links that point to product pages and that share uniform HTML format layout.

[2]These services are used to support search engine optimization (SEO).

2.3 Identifier Extraction

The last phase of source discovery has the objective of extracting product identifiers from the pages collected by the crawler. These identifiers will be used to launch new searches for discovering other sources.

To extract the product identifiers that can feed a new search, we exploit the local homogeneity of web sources. For each discovered source, we use the set of known identifiers to automatically annotate the pages collected by the crawler; then, for each annotation that exactly matches a textual leaf node of the DOM tree associated with the page, we infer an extraction rule (e.g. an XPath expression). To this end, we use the technique proposed by Dalvi *et al.* [2] to infer the extraction rules given noisy training data, that is, pages annotated by possibly erroneous labels. Because of the local structural homogeneity of the sources, we expect that the application of these rules returns the identifiers that are present in the pages.

3 Data Discovery and Extraction

We now describe our strategies to automatically extract product descriptions from the collections of product pages harvested by the previous task.

We have concentrated our efforts on specifications, in the form of pairs of attribute name and value, and on the product identifier. Future investigations will target our attention to extract and process also price and product reviews. We now describe our approach to extract specifications, and in Section 4 we illustrate our technique to extract product identifiers associated with the product on a page.

Automatically extracting specifications in many websites for different categories of products is challenging. Each website adopts a local template to generate the product specification pages. Then, to accurately extract the data, one should generate a specific wrapper for every website. A solution to this problem could be that of applying an automatic wrapper inference technique, such as Roadrunner [1]. However such a solution exhibits two major drawbacks: first, it has limited performance with irregular templates; second, it extracts a lot of meaningless data, since it considers every item that is not part of the template as data to extract. Another approach is to develop wrappers leveraging domain knowledge, as in [7]. However, as we already discussed, this would limit the opportunity of discovering *tail attributes*; also, with a large number of product categories this approach would require significant effort because of the heterogeneity of specifications across categories.

We have adopted a solution that represents a good trade-off between the above approaches. On the one hand, we exploit publication practices that occur globally in many product websites. On the other hand, we leverage the local homogeneity exhibited in large websites, which build pages according to a bunch of local templates.

Our solution splits the problem in two separate phases: *data discovery*, i.e., detecting the portion of product pages that contain the specifications, and *data extraction*, i.e., extraction of the specification, as attribute name and value pairs.

3.1 Data Discovery

We have observed that although specifications can be contained in different HTML structures, they are primarily found within HTML table and list elements. By manually inspecting 301 pages from a great variety of products, we have noticed that 62% of the specifications were inside an HTML table element, and 31% were inside an HTML list element; the remaining 7% were in other miscellaneous HTML elements.

Web pages may contain many tables and lists; however, we have noticed that, independent of product category and of the site, product pages exhibit structural characteristics that can be exploited to determine if a table or a list contains the product specifications (as opposed to being used, e.g., only for formatting purposes). Therefore, we have trained a classifier to detect tables and lists that contain product specifications. We used a Naive Bayes classifier, considering features dealing with table and list contents, such as, statistics about number of links, number of tokens in the leaf nodes, depth of the leaf nodes, etc. (see [12]).

3.2 Data Extraction

In order to extract attribute name and value pairs, we have considered two strategies.

The first strategy adopts a simple heuristic based on the observation that the structure of the fragments containing the specifications in tables and lists are very homogeneous, even across sources. By inspecting these tables and lists, we determined that attribute-value pairs of the specification are often contained in the html row element and that the first and second text elements of each row represent the attribute name and value, respectively. A similar heuristic is applied for the elements of lists classified as specifications.

The second strategy, which is applied on tables, uses the same technique adopted to extract the identifiers, described in Section 2.3. In this case, attribute names and values from the input sample pages are used to annotate the pages of the discovered sources. From the annotations we infer extraction rules. To obtain a more effective approach, we generalize these extraction rules to work on all the rows of the table, thus extracting all the attributes that it offers, including those that are not in the input set.

4 Data Linkage

Our approach to data linkage for product specification pages exploits the opportunity that product pages usually contain a product identifier. However locating the product identifiers in product pages at Web scale is quite challenging:

- It is not easy to locate, within the HTML of the product specification pages, the string that represents the identifier; some sources adopt microdata markups (such as schema.org), but their diffusion is limited [10]. Usually identifiers consist of a single token that, for some categories of products, follow specific patterns. But at Web scale, it is not possible to generalize these patterns (as done, for instance, in [9], which concentrated on a handful of sources), because of the large variety of patterns. Similarly, it is not possible to focus only on specific patterns, e.g., those associated with popular standards (as done, for instance, in [13]) because of the skewed distribution of the patterns. To give a concrete example, we have analyzed the identifiers extracted from the subset of pages annotated with microdata markups in the collection of sources discovered by our pipeline. We observed 930 different patterns for $33,281$ values, with the most frequent pattern (a sequence of 8 digits) matching less than 23% of values; the most frequent pattern associated to a standard was GTIN-13, with frequency 3%.

- Product pages usually contain identifiers not only for the main product presented in the page, but also for related products (e.g., suggested products, products bought by other users, sponsored products, etc.).

- Some identifiers may only be local (i.e., only within a source), not useful for linkage across sources. Local identifiers from different sources may conflict; similarly, conflicts may occur among global identifiers of products from different categories. Hence, different product pages associated with the same identifier are not guaranteed to refer to the same product.

To overcome these challenges, we leverage the redundancy of information that occurs at the global level and the regularities of the sources and uniqueness of information that occur at the individual source level [11]. Our approach to data linkage for product pages consists of an iterative phase to extract and filter identifiers, and a conclusive phase to resolve conflicting identifiers and generate the linkages.

4.1 Identifiers Extraction and Filtering

We start from a *seed set* of high quality product identifiers, which are used as keywords for searching among the discovered sources product pages that might refer to these products.

Next, for every retrieved product page, we infer an extraction rule for every HTML region containing an occurrence of the searched identifiers. To infer the extraction rule we use again the technique based on noisy annotations. However, here we consider as worth extracting also regions that contain the identifiers, not only those that exactly match the identifiers. In fact, for the purpose of linkage we are interested to extract the identifier that corresponds to the product on the page, and in many sources this is in a textual node, together with other tokens.

Based on the assumption of the local regularities of the sources, the extraction rules are used to obtain regions containing identifiers from all the other product pages in the same source. From every region returned by the rules, we have to select the token that represents a candidate identifier for the primary product of the corresponding page.

Since we cannot rely on a set of predefined patterns to select identifiers among the tokens of large regions, we consider that usually a source provides at most one page for a product. Therefore, we take the frequency of the tokens as a measure of their ability to serve as identifiers.

An immediate idea is to consider the token source frequency, that is, the token with the smallest number of occurrences within the source. However, considering every source separately does not work well when searching for global identifiers because many sources, especially tail sources, contain tokens that are locally rare, but do not represent global identifiers. For example, consider a small source where there is just one laptop with a touchscreen: if the keyword `touchscreen` is used along with the description of the product, it would be very rare at the source level, and thus it could be erroneously regarded as a good candidate identifier. Even if these tokens might appear as very selective at local level, they are much more frequent if considered globally, especially in the head sources. Continuing the above example, `touchscreen` is a pretty frequent token at the global level.

Therefore, we consider the token collection frequency, defined as the total number of occurrences of a token in the whole collection. In this way, we take into account both local and global characteristics. It is worth noting that because of this property we can perform data linkage only once we have gathered a large number of sources.

The above extraction and selection processes may produce incorrect identifiers, for many reasons: a wrong selection of the candidate identifiers; a region returned by an inaccurate extractor; the presence of regions containing identifiers that do not refer to the main product of the page (e.g., suggested products). To improve the precision, we consider the duality between good extractors and correct identifiers: an extractor is good if the identifiers of its regions are correct; similarly, an identifier is correct if it comes from a region returned by a good extractor.

The selected identifiers are then iteratively used to trigger another search on the source dataset.

4.2 Resolution of Conflicting Identifiers

Due to the variety of information across sources, and the presence of local and global identifiers, at the conclusion of the iterations different products could share the same identifier across sources. To improve the accuracy of linkage, we identify these conflicting identifiers by considering that every source consists of a homogeneous set of products: although the criteria that define the uniformity of the product categories are local, not global, with a large enough dataset it is likely that many pairs of products co-occur in many sources because they adopt similar (even if they are not identical) categories. Then, we consider identifiers that co-occur with multiple identifiers in many sources more reliable for the purpose of linking.

5 Lessons Learned and Future Work

In an experimental evaluation performed between Sept 2014 and Feb 2015, we have trained the focused Dexter crawler [12] to gather product pages from 10 coarse categories: camera, cutlery, headphone, monitor, notebook,

shoes, software, sunglasses, toilet accessories, televisions. The crawler discovered 3.5k websites, for a total of 1.9M pages. Each website contributed to provide pages for the different categories, and pages were grouped into $7,145$ clusters, corresponding to the local categories exposed by the websites (on average every websites has 2 local categories). The dataset is publicly available on-line.[3]

Building a Benchmark Product Dataset We compared the contents of our dataset with pages in Common Crawl,[4] an open repository of web crawl data. About 68% of the sources discovered by our approach were not present in Common Crawl. Only 20% of our sources contained fewer pages than the same sources in Common Crawl, and a very small fraction of the pages in these sources were product pages: on a sample set of 12 websites where Common Crawl presented more pages than in our dataset, we evaluated that only 0.8% of the pages were product pages.

These results suggest the critical need for the community to build a suitable benchmark product dataset to conduct big data research. Our dataset can be considered a first step in this direction. Another important step would be that of maintaining the dataset over time, as discussed next.

Maintaining the Benchmark Product Dataset: Addressing the Velocity Challenge In March 2018, we have checked all the URLs of the pages of the dataset. We have observed that just 30% of the original pages and 37% of the original sources are still valid (we consider a source valid if it contains at least one working URL). We also performed an extraction of the product specifications. We obtained complete specification from just 20% of the pages.

These numbers clearly indicate that the velocity dimension affects all the tasks of the pipeline. Developing solutions to collect snapshots over regular time intervals and perform data integration over time can open intriguing research directions. While some activities, such as checking the appearance/disappearance of sources can be done on monthly basis, others, such as crawling websites to check appearance/disappearance of pages and changes in the pages should be performed more frequently. To this end, the development of efficient incremental solutions for source discovery and web crawling represent interesting research directions.

As our experiments emphasize, data extraction rules are brittle over time. The development of wrappers resilient to changes in the pages has always been a primary goal in data extraction research. A dataset with multiple snapshots over a long interval of time, as the one that we have advocated above, could serve as a benchmark for data extraction solutions.

Harnessing Velocity We observe that while velocity represents a challenge, it could also become an opportunity. For example we observe that analyzing changes in the pages could help improve our data extraction and data linkage techniques. For example, examining the same product page over time may help us to more easily separate out information about related and suggested products, since those may change faster over time than the descriptions of the product in the page.

Schema Alignment We are currently working on the development of techniques to perform schema alignment for our product domain. The main difficulties that we have to face are due to the heterogeneity at the schema and at the instance level, due to the large number of independent sources.

To give a concrete example of the heterogeneity at the schema level, consider the dataset collected using the Dexter crawler, described earlier in this section. The specifications extracted from these sources contain more than $86k$ distinct attribute names (after normalization by lowercasing and removal of non alpha-numeric characters). Most of the attribute names (about $85k$) are present in less than 3% of the sources, while only 80 attribute names occur in 10% of the sources, with the most popular attribute name occurring in just 38% sources.

[3] Note that the on-line version (`https://github.com/disheng/DEXTER`) is an extension of the dataset presented in [12].

[4] `http://commoncrawl.org/`

The solution that we are investigating exploits data linkage to cluster attributes that share the same values. Since heterogeneity occurs also at the instance level, we aim at progressively finding correspondences between the clusters resolving increasingly complex matches between values with different representations.

Addressing the Veracity Challenge and Data Fusion Analyzing the sources of our dataset we noticed that some clusters contain product pages from different categories. In some cases, the errors are in the sources: some websites adopt unexpected (or even completely wrong) criteria as, for example, classifying monitors under a laptop category or vice-versa; other websites aggregate products and related accessories (which represent another category of products). In other cases, the errors are due to wrong classifications by the system.

To overcome this issues we want to investigate different solutions. First, we believe that introducing feedbacks in our pipeline could significantly improve data quality, especially for precision. For example, alternating source discovery and data linkage we could select better identifiers to feed to Search for source discovery, thus increasing the precision, with respect to the target category, of the sources. Similarly, results from schema alignment could help improve the precision of linkage, as pages whose attributes do not align are unlikely to represent the same product. Another promising direction to improve precision without penalizing recall is to study solutions to exploit humans in the loop. In particular, we aim at developing and evaluating techniques based on active learning and crowdsourcing to continuously train the classifiers with effective and updated training sets.

We have observed many inconsistencies between values of a product attribute across sources. Existing data fusion techniques [4, 5] can help to resolve these inconsistencies when they are due to honest mistakes, possibly in combination with extraction errors. However, the product domain also exhibits inconsistencies due to deceit, where sources may deliberately provide imprecise or erroneous product characteristics. Identifying and effectively addressing such inconsistencies at web scale is an important direction of future work.

Beyond Source Homogeneity Our approach to data extraction is based on the assumption that pages are structurally homogeneous at the local source level. This is a valid assumption for a vast majority of websites, but there are exceptions. For example, some websites that publish used products have a weak template and leave the seller the freedom to publish the product specification without any structural imposition. For some application scenarios, one can simply drop these sources. If on the contrary they are important, data extraction should be performed by solutions that do not rely on the template.

Beyond Product Specifications So far we have considered the extraction of the identifiers and of the specifications. However important data that complete the product description are price and reviews. Challenging issues for the extraction of price are to distinguish the price of the principal product in the page from the prices of other products, such as suggested product, similar products, and the actual price from discounts or list price. Reviews represent important information in many applications. An interesting problem is how to combine structured data from the specification with the unstructured data of the reviews.

References

[1] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1-2):21–52, 2008.

[2] N. Dalvi, R. Kumar, and M. Soliman. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4(4):219–230, 2011.

[3] N. Dalvi, A. Machanavajjhala, and B. Pang. An analysis of structured data on the web. *Proceedings of the VLDB Endowment*, 5(7):680–691, 2012.

[4] X. L. Dong. How far are we from collecting the knowledge in the world? In *Keynote at 19th International Workshop on Web and Databases*. ACM, 2016.

[5] X. L. Dong and D. Srivastava. *Big data integration*, volume 7. Morgan & Claypool Publishers, 2015.

[6] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[7] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang. Diadem: thousands of websites to a single database. *Proceedings of the VLDB Endowment*, 7(14):1845–1856, 2014.

[8] J. Jiang, X. Song, N. Yu, and C.-Y. Lin. Focus: learning to crawl web forums. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6):1293–1306, 2013.

[9] H. Köpcke, A. Thor, S. Thomas, and E. Rahm. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 545–550. ACM, 2012.

[10] R. Meusel, P. Petrovski, and C. Bizer. The webdatacommons microdata, rdfa and microformat dataset series. In *International Semantic Web Conference*, pages 277–292. Springer, 2014.

[11] D. Qiu, L. Barbosa, V. Crescenzi, P. Merialdo, and D. Srivastava. Big data linkage for product specification pages. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of data*. ACM, 2018.

[12] D. Qiu, L. Barbosa, X. L. Dong, Y. Shen, and D. Srivastava. Dexter: large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment*, 8(13):2194–2205, 2015.

[13] A. Talaika, J. Biega, A. Amarilli, and F. M. Suchanek. Ibex: harvesting entities from the web using unique identifiers. In *Proceedings of the 18th International Workshop on Web and Databases*, pages 13–19. ACM, 2015.