OpenTRIAGE: Entity Linkage for Detail Webpages

Roger Voyat[1], Valter Crescenzi[1] and Paolo Merialdo[1]

[1]*Department of Engineering – Università Roma Tre, Via della Vasca Navale 79, Rome Italy*

Abstract

We present OpenTriage, a system for extracting structured entities from detail Web pages of several sites and finding linkages between the extracted data. The system builds an integrated knowledge base by leveraging the redundancy of information with an Open Information Extraction approach: it incrementally processes all the available pages while discovering new attributes. It is based on a hybrid human-machine learning technique that targets a desired quality level. After two preliminary tasks, i.e., blocking and extraction, OpenTriage interleaves two integration tasks, i.e., linkage, and matching, while managing the uncertainty by means of very simple questions that are posed to an external oracle.

1. Introduction and Overview

Although the amount of data that are available on the Web is growing with an exponentially pace, most of these data cannot be directly processed by applications, as they are published in HTML, a format meant to be displayed by a Web browser and consumed by humans rather than processed by machines. A lot of those data are published by mean of pages such as those depicted in Figure 1: detail pages [1, 2] that report data about one real world entity, e.g., the italian basketball player Danilo Gallinari. Several sites may independently publish data about the same real world topic entity, i.e., Figure 1 shows two pages taken from two sport sites (espn.com and foxsport.com).

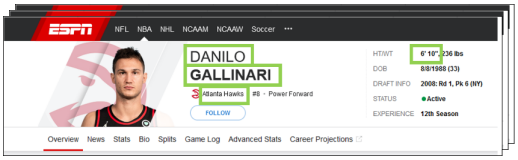
Websites may publish different attributes, and detail pages about a different set of entities. Therefore, the integration of partially overlapping information coming from several sites is a challenging, yet rewarding, problem of building knowledge bases for several popular domains. Data are found in detail pages published on the Web and extracted into structured records such those shown in Figure 1c and 1d before being integrated.

The problem of finding whether two attributes of two distinct sources actually have the same semantics is commonly known as schema matching [3, 4, 5] (for example the pair of attributes (a_2^1, a_2^2) in Figure 1c and 1d), while the problem of finding which structured records relate to the same real world entity is often called entity linkage [6, 7, 8, 9].

Challenges & Opportunities Although linkage and matching are both considered difficult Big data integration problems [10], in many practical settings, including ours, the former is more often considered the most challenging one.

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy

roger.voyat@uniroma3.it (R. Voyat); valter.crescenzi@uniroma3.it (V. Crescenzi); paolo.merialdo@uniroma3.it (P. Merialdo)



(a) i site espn.com.

(b) ii site foxsport.com.

	a_1^1	a_2^1	a_3^1	a_4^1
o_1^1	Danilo	Gallinari	Atlanta Hawks	6' 10"
o_2^1	Clint	Capela	Atlanta Hawks	6' 08"
...	...	...	...	...

(c) Extracted records from espn.com.

	a_1^2	a_2^2	a_3^2
o_1^2	Danilo	Gallinari	Atlanta Hawks
o_2^2	Trae	Young	Atlanta Hawks
...	...	...	...

(d) Extracted records from foxsport.com.

Figure 1: Running example: Web detail pages and data extracted from *NBA player* domain.

First, for the dimension of the problem, as the number of input entities published in a Web source can outgrown the number of their attributes by several order of magnitudes: a site can publish thousands of detail pages with no more than a few dozen different attributes. One might argue that the number of attributes scales proportionally to the number of Web sources, but in practice, there are a few head attributes that are used by most of the sources [11], giving rise to skewed distributions. Second, the challenges posed by the matching problem directly affect the linkage one, as well: to decide whether two records are related to the same entity, it is important, especially in the presence of overlap between the entities of two sources, knowing the matching of the attributes of the compared records.

Another challenge is that two matching values can have formats and semantics that vary from source to source, or even from subgroup of entities to subgroup of entities within the same source. For example, the postal code in England is an alphanumeric string locating a single building, while in Italy the postal code is a numeric string often associated with several blocks of a quite big city; in another domain, a smartphone manufacture might adopt a naming policy for its models that distinguish a model from another for a single letter, e.g., *Apple IPhone X* and *Apple IPhone XS*, whereas another might adopt totally different names, e.g. *OPPO Find X5* and *OPPO Reno 6*.

Although automatically building an integrated knowledge base from detail pages published by many sources is a challenging problem, the are interesting opportunities as well: especially for the most popular domains, the Web is an almost inexhaustible source of detail pages with redundant information about the same entities, to the extent that we can assume that our source base is redundant enough to contain at least two detail pages and at least two values for each domain entity and every attribute of the knowledge base we want to create.

Problem Description OpenTriage takes as input a domain D made of a set of sources $\mathcal{S} = \{S_1, S_2, \ldots\}$ where each source S_i is a set of n_i detail pages each publishing information about a single entity from a set of domain entities $\mathcal{O} = \{o_1, o_2, \ldots\}$. A domain entity o_i includes a set of domain attributes $\mathcal{A} = \{a_1, a_2, \ldots\}$ whose values can be populated with values coming from the HTML source code of the corresponding detail page by means of a set of extraction rules associated with the domain attributes.

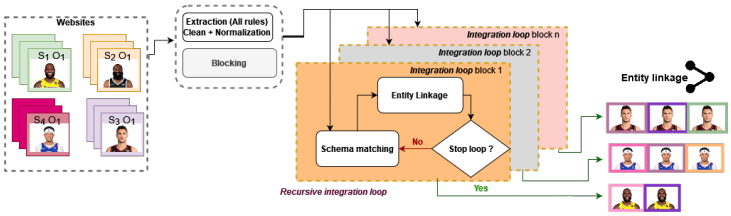


Figure 2: OpenTriage high-level architecture.

In the following, we introduce OpenTriage, a Human-in-the-Loop system that produces as output a set of entity linkages, i.e., pairs of pages related to the same domain entity, and a set of attributes matches, i.e., pairs of extraction rules related to the same domain attribute. Indeed, OpenTriage also generates the extraction rule, if any, that locates the value of domain attribute from the HTML source of any of the detail pages from a source.

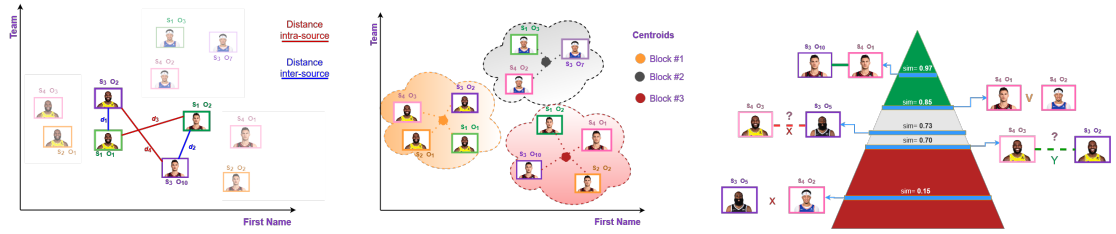
Figure 1 shows two detail pages that include several attributes (highlighted in green). Figure 1c and Figure 1d show the extracted records with some anonymous attributes such as a_1^1 , a_2^1, a_3^1, and a_4^1 (which is the Height). Possible matches are, for example: (a_1^1, a_1^2) (i.e., First Name), (a_2^1, a_2^2) (i.e., Last Name), $a(_3^1, a_3^2)$ (i.e., Team); a linkage is (o_1^1, o_1^2) which is a pair of entities from detail pages associated with Danilo Gallinari.

2. OpenTriage System

OpenTriage orchestrates several integration tasks as shown with the high-level architecture shown in Figure 2. A data extraction algorithm is applied over every input page to generate a set of extraction rules for every source, thus associating each page with a record of extracted data. Simultaneously, an initial blocking phase will group pages into blocks of pages potentially in linkage, with the goal of never separating two pages about the same entity into two different blocks. Then, OpenTriage launches a new integration task over every block of detail pages, as shown in Figure 2: iteratively, a linkage task and an attribute matching task are interleaved so that the produced linkages are used to discover attributes matches which, in turn, are used to provide better linkages. This process can be supported by an external oracle that contributes to halt the iteration when the desired quality levels are achieved.

Extraction rules generation and blocking OpenTriage generates a set of candidate extraction rules for every input page and associates each detail page with a record of extracted data. This task is based on a state-of-the-art unsupervised algorithm that works per site. It is beyond the scope of the present paper as it's already extensively discussed in previous works [12, 13, 14]. As already shown in [2], by assuming that the set of candidate extraction rules is complete, i.e., it includes at least one correct rule, the following integration tasks will always prefer the correct rules over the noisy ones, which do not find values that match with values from other sites. In the following we will not consider the presence of such noisy rules to simplify the discussion.

Then OpenTriage use a blocking algorithm working only with the titles of the input pages. The page title is easy to extract from HTML and, with rare exceptions, it contains some



(a) Entity distance (b) Blocking result (c) Block pyramid

Figure 3: Entity distance, block & Pyramid:Figure 3a shows entity alignment with sim. distance inter (d_1, d_2) and intra-source(d_3 and d_4). In Figure 3b is shown a block result and Figure 3c depict a block pyramid with green (pairs in linkage), gray (the uncertainty) and red zone (all remaining non-linkage pairs) used for integration.

information about the entity described in a detail page. More precisely, in blocking, we compute the distance between two pages (Figure 3a) as the inverse of the Jaccard-index (J) between the sets composed by the tokens contained in their HTML titles. $J(t_1, t_2) = \frac{|t_1 \cup t_2|}{|t_1 \cap t_2|}$. All pairs of pages whose similarity is greater than a threshold are considered as belonging to the same block. We empirically set this threshold to a value (0.6) that our experiments showed to be low enough to have an almost perfect recall, even if that means sacrificing the precision, so that pages in linkage will belong to the same group.

With blocks generated, OpenTriage may obtain a first approximate noisy linkage. Since the number of input pages can be too big for a traditional quadratic linkage algorithm, as it is the case for many other linkage solutions [15, 16], we use a basic blocking algorithm [17, 18, 19] that allows to linearly group the pages into block of potentially similar detail pages and later we confine only within the pages of a single block further, and more expensive, elaborations. As later we won't look for linkages across distinct blocks, blocking techniques are supposed to not split detail pages related to the same entity in different block to prevent false negatives, yet they should spread the entities in a reasonable number of not-too-big blocks as shown in Figure 3b.

Once the bootstrapping phase is over, one record of extracted data is associated with each input detail page, and the pages have been clustered into blocks of pages potentially related to the same real world entity. This is the input for the following processing stage.

Linkage of extracted records After the bootstrap phase generates blocks of pages potentially related to the same topic entity, for each block OpenTriage performs a data integration task that aims at computing both the linkages and the matches over the records of the pages in a block. The data integration task relies on a normalized distance function between pairs of records with known matches for finding the linkages, and symmetrically it leverages a distance function between pairs of attributes with known linkages for finding the matches. For the sake of presentation, in the following we start assuming that the matches are given, so that we can focus on finding the linkages, which usually is the most challenging integration task. Later, we will show how we can exploit a seed set of (possibly noisy) candidate linkages to compute an instance-based distance measure between attributes to solve the attribute matching problems, as well. That explains why an entity linkage task and an attribute matching task are interleaved

in Figure 5.

We compute the normalized distance for every possible pairs of pages in a block, and we rank these candidate linkages by distance. These results are organized in a data structure, the Block Pyramid – BP, that we depicted in Figure 3c, which is at the base of the following operations.

Triage OpenTriage scans every BP to perform a kind of triage over pairs of pages belonging to the same block. For an ideal domain, where the distance is capable of perfectly ranking the candidate linkages, all the pairs that correspond to actual linkages are at the top of the BP (green zone in Figure 3c) whereas all the other pairs (representing pairs of pages that are not in linkage) should be at the bottom of the pyramid (red zone): a single threshold on the distance can perfectly separate the green zone from the red one.

We can further develop this characterization of a domain in term of BP by introducing a domain property called Domain Separability (DS). A domain is separable if the distance between two entities in linkage is always smaller that distance between any pair of entities that are not linkage. Unfortunately, this property does not hold in practice as a distance function cannot always separate linkage and non-linkage pairs: as shown Figure 3c, there is a "gray" zone that includes pairs that are in linkage but their distance is greater than the distance of pairs that are not in linkage, or vice-versa.

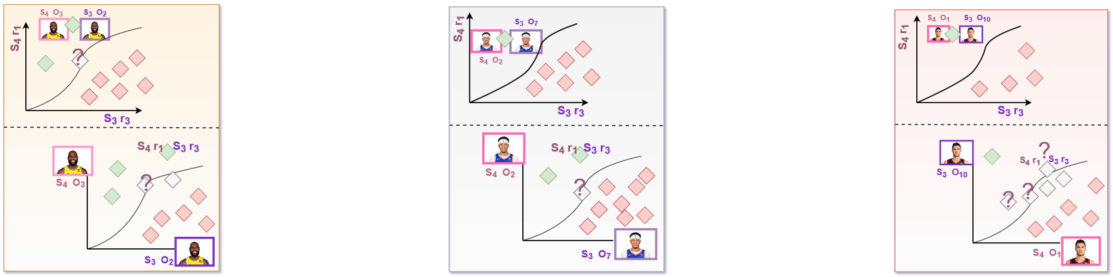
The gray zone confirms the presence of DS violations and the possibility that a non-linkage could have a lower distance than an actual linkage. Our experimental evaluation with real pages has shown that the BPs have different size and "shape" not only across distinct domains, but also from block to block within the same domain. Every page and every attribute can exhibit, somehow, characteristics local to the Web source, or even to the block, for several reasons, including: First, the distances are affected by the formats and unit of measure of the data adopted by a source; second, there are sources publishing inaccurate, false, or just obsolete information, and finally, there might be other minor semantics differences between published attributes which are really challenging to detect. For example, an address can be published with any of the following values: "1600 Amphitheatre Pkwy, Mountain View, CA 94043, United States" ; "1600 Amphitheatre Parkway in Mountain View, California (USA)".

Automatically discovering the best triage for every block of pages is therefore a challenging and difficult task, that we approach by looking for two thresholds (the green-to-red threshold separating the green zone from the gray one, and the gray-to-red threshold) and by making an additional assumption holding for many interesting real domains on the Web.

We say that a Web source exhibit the Local Consistency property (LC) if it does not publish more than one detail page related to the same real world entity. This means that the domain does not include linkages of pages from the very same source, which is often the case. Otherwise, it would mean that source is unnecessarily publishing twice or more information about the same entity with the risk of introducing inconsistencies.

Assuming the LC, as shown in Figure 3c, OpenTriage can find the first green-to-gray threshold as the distance of the first non-linkage pair under the green zone. The pair can be located by analyzing BP from top, as that having the smallest distance and made of pages belonging to the same source.

To identify the second gray-to-red threshold and to complete the triage, OpenTriage lever-



(a) Block 1 duality (b) Block 2 duality (c) Block 3 duality

Figure 4: An example over the *NBA* domain. The top Cartesian planes in sketches the linkages finding problem: the points correspond to candidate linkages and their distance is computed by considering the values of two matching attributes associated with the axes. The Cartesian planes in the bottom of the figures illustrate the matches finding problem: they contain points corresponding candidate matches and their distance is computed by considering the values of those attributes of two entities associated with the axes. The gray pairs are those violating the domain separability.

ages the feedback from an oracle, that in practice can be approximated by means of crowdsourcing techniques [20, 21, 22].

2.1. Human-in-the-Loop

OpenTriage exploits an external oracle to perform an approximate triage for each BP. It poses simple binary questions that double check whether two structured records refer to the same entity. An example of the questions posed is: *Do the records "(Danilo Gallinari,Atalanta Hawks,6'10 ' ') " and "(Danilo Gallinari,Atalanta Hawks)" refer to the same real world entity?*

Our algorithm processes the candidate pairs in a BP ordered by decreasing distance, and starts posing query only for pairs coming after the first green-to-gray threshold: The goal is that of finding (if any) a pair of records in linkage with the smallest distance that better approximates the gray-to-red threshold. To avoid posing too many questions that just collect useless negative answers, it uses two stop conditions: (i) halt when it receives too many consecutive negative responses, (ii) halt when the next pair is too far from the last one. Both conditions depend on OpenTriage configuration parameters. The best quality-to-cost ratio in our experiments is found with this setting: we halt after 5 consecutive negative answers and with a pair which is at least a 0.2 distance from the last pair. The gray-to-red threshold is then approximated by the smallest distance of the pairs in linkage discovered by the oracle. In case none linkage has been found, it is assumed that gray zone shrinks to an empty one and the two thresholds collapses.

2.2. Matches/Linkages Duality

OpenTriage leverages a distance measure between two records for finding the linkages between pairs of pages in the same block: that distance is computed by assuming that the attribute matches of the involved records are known, and then averaging over the distance of the values of the matching attributes. To relax this assumption, which is not holding for our input made just of records automatically extracted from detail Web pages and having anonymous labels, we also

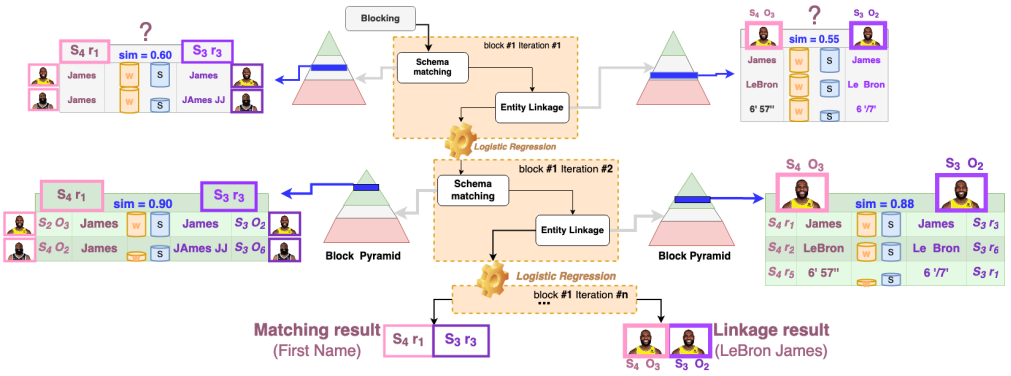


Figure 5: Integration tasks over the one BP from the *NBA* domain.

need to find the attribute matches. OpenTriage tackles the problem by using an instance-based distance measure between attributes from two sources: it is computed by averaging over every pairs of attribute values from records in linkages, i.e., from entities belonging to the overlap of the sources.[1]

OpenTriage finds the matches by framing the finding problem as a the dual problem of the linkages finding problem. The dual integration task leverages a dual BP in which all the possible matches are ranked according to their instance-based distance measure. These, in turn, are based on the linkages obtained by a previous execution of the primal task. The dual triage will classify the top candidate attribute matches in the green zone by exploiting the dual property of the LC, i.e., the same source never publishes twice the same attribute, and of the DS, i.e., the instance-based distance between two matching attributes is always smaller that distance between two non-matching attributes. The dual triage will exploit an oracle to approximate the gray-to-red threshold by posing very simple binary questions such as: "are 'Danilo Gallinari' and 'D.Gallinari' values of two matching attributes?".

Figure 5 and Figure 2 show that the primal linkage task and the corresponding dual matching task are interleaved. Each task produces results consumed in a following iteration of the other task. OpenTriage's main processing loop starts with all the weight parameters used in the normalized distance functions having the same initial values. The linkages/matches produced by every triage are then used to update the weights by means of a statistic linear model (i.e., logistic regression) that also considers the answers provided by the oracle. Updating weights implies a possible reshaping of the BP (some pair might change "color"): the iterations are halted as soon as the results do not change too much, usually just after 2 or 3 iterations. Figures 4a and 4b show a block with a few pairs just on, or close to, the border separating positive linkages/matching pairs from negative ones. Those pairs might need a specific oracle question to be fixed after a triage wrongly classified them. Figure 4c shows an even larger number of questions that might be asked, as the block includes many pairs close to the line separating the positive linkages from the negative linkages.

[1]This is yet another frequent reason for the existence of the gray zones in the matches BPs: if the overlap is too small, an instance-based distance function might become not reliable enough to provide good distance values.

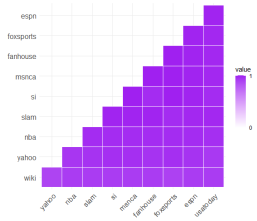
3. Experimental Results

OpenTriage discovers extraction rules, attribute matches and entity linkages by interleaving linkage and matching tasks while posing questions to an oracle. Those questions are constrained by an overall budget, i.e., the maximum number of questions that can be posed to solve a whole input domain, and represents the cost measure for our approach. As explained in Section 2.1, OpenTriage adopts two halting conditions that avoid consuming too many questions for the same block. Our preliminary experiments show that the size of the gray zone as estimated by the block triage procedure, monotonically shrinks during the consecutive iterations of OpenTriage integration task over that block. Depending on the characteristics of the input domain, OpenTriage can be used both as a stand-alone technique to fully integrate a domain, or as an approach to actively isolate the best samples of a training set. This is later used to feed a machine learning algorithm. Indeed, for easily separable domains, OpenTriage can help to find and fix a few outliers that need a small number of questions to the oracle. For these domains, it can automatically separate most of the positive pairs by means of the triage procedure. Conversely, for a highly non-separable domain, it can be better used to select non trivial pairs that stand in the gray zone. The questions to the oracle are saved for highly informative samples that can be used to train a non-linear machine learning algorithm.

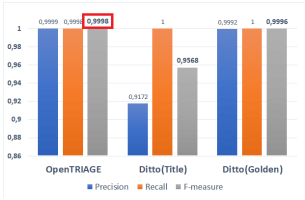
Setup As we are not aware of any other approach as ours that tackles three integration problems over detail Web pages at the same time (extraction, linkage, and matching), for a preliminary experimental evaluation we compare our system with an entity resolution state-of-the-art approach over Web data. Namely, we consider Ditto [23] over the SWDE [24] benchmark. We focus on 2 out of 8 domains available SWDE: NBA players and University. They have rather different and interesting characteristics: the NBA players dataset is composed of $4.5k$ pages from 10 websites, with large overlap of entities across sites as shown in Figure 6a(i.e., $8.8k$ of entity linkages); on the contrary, the University dataset includes $20k$ pages from 10 sites, with a rather low level of redundancy across sources as shown in Figure 6d(i.e., $4.3k$ entity linkages). For each processed domain, several executions were made with an increasing number of of questions to the oracle (our cost budget). The blocking task for the NBA domain has generated 90 blocks, whereas 1060 blocks were generated for the University domain. The former domain has a rather higher level of redundancy than the latter: OpenTriage ends up consuming a much smaller number of questions (80 vs 1350) by exploiting the available redundancy.

With or Without DITTO Ditto is a deep learning Entity Resolution system based on pre-built language models but it does not solve neither the extraction problem, nor the matching problem. We considered two settings for comparing our linkage results to those produced by Ditto.

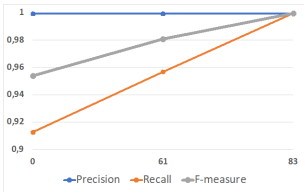
- Ditto-Title, we use the page title as a textual attribute describing the entity contained in the page. The title can be trivially extracted and it is useful for linkage as its value is usually strictly related to the topic entity. For example, in the NBA page the title contains the name of the player, and in the University page it contains the name of the university.



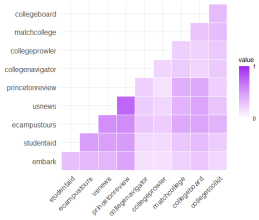
(a) NBA entity overlap



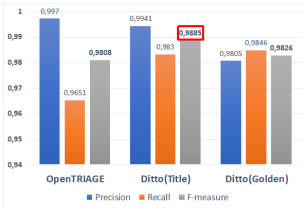
(b) NBA linkages



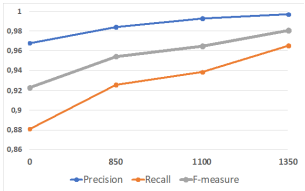
(c) NBA budget



(d) University entity overlap



(e) University linkages



(f) University budget

Figure 6: Domain Overlap & results: Figures 6a and 6d show that domains have different overlap on entity published across websites. Figures 6b and 6e the experimental P, R, F of linkages for OpenTriage, Ditto-Title and Ditto-Golden. Figures 6c and 6f the quality of output linkage w.r.t. budget spent.

- Ditto-Golden considers as input the golden structured records provided with the SWDE benchmark: these records correspond to data extracted by manually defined extraction rules. Observe that, in a real scenario, for using Ditto one would need an additional effort for crafting the extraction rules.

Ditto is a supervised system: we have trained it using all the linkages obtained by OpenTriage, and then we measured the quality of the output linkages w.r.t. a golden set of linkages using standard quality metrics. Precision (P) is the percentage of output linkages which are correct, Recall (R) is the percentage of linkages in the golden which are in the output linkages, and F-measure is their harmonic mean. Figures 6b and 6e show the results OpenTriage achieved for the considered domains with unlimited budget. In both the domains the output quality is good, but the University domain required far more questions than the NBA domain ($1,350$ vs 83). As long as the input Web sources have enough overlap in the set of provided detail pages, our approach adapts well to the characteristics of the input domain even if this means paying for many additional questions. Figures 6c and 6f show how the quality of the results is affected by a limited budget. We observed a stronger impact on the recall than on the precision, especially for domains with large overlap, where a few additional questions can significantly improved the triage quality.

For domains with a limited overlap of entities and attributes, the amount of available redundancy might result too small to support an effective triage. In this situation, OpenTriage would end up generating far more queries supporting the system iterations than it does in presence of a good amount of redundancy. As a future work, to solve this problem, a possible natural evolution for OpenTriage is that of supporting the incremental integration of new sources with different degrees of overlap. Indeed, a new source would increase the size of the problem,

but at the same time it might even more incisively contribute to lower the overall costs if it significantly overlaps with the other sources already available for the domain [25].

4. Related work

Big Data Integration [10] and Information Extraction [1] are broad research topics related to our approach: We only have the room for providing some entry points to such a vast literature.

The three main problems we study in this proposal, i.e, data extraction [12, 26], schema matching [3, 4, 5], and entity linkage [6, 7, 8, 9], have been most often tackled in isolation, we some rare exception [2]. However, entity linkage on Web data has received much less attention so far. Blocking techniques for entity resolution is a topic often been studied in the literature [16, 15]. Unlike the Closed Information Extraction approaches [27, 28] where the managed knowledge base does not grow in terms of subjects and predicates but only in terms of values and data, OpenTriage falls in the realm of the Open Information Extraction approaches as it finds new domain attributes while processing the input sources. The research community has pursued the problem with different types of solutions. Traditional approaches [2, 13, 29, 30, 28, 31, 32, 33, 34] take advantage of the regularity of the HTML structures for pages published by templated websites. Other recent proposals [1, 35] use an hybrid approach exploiting both the regularity of the publishing patterns for HTML pages, and the tagging of attribute values provided by human workers. The main difference with OpenTriage is that our system do not require a partially populated knowledge base to trigger the acquisition of new data.

Other information extraction works [36, 37, 24] have exploited computer vision methods. For example they also consider page screenshots, the bounding boxes around the text nodes, and their visual distances. The latter requires expensive rendering for data processing and need a lot of training data for ML models. A recent approach is FreeDOM [38], that avoids rendering costs still need human feedback for tagging attributes values. Unlike the latter [38], OpenTriage does not need expert crowdsourcing workers.

Conclusions & Future work OpenTriage can be used either directly as an entity linkage solution over Web pages, or as an active learning algorithm for selecting the most profitable training data to feed a following non-linear ML solutions like Ditto [23]. Our future work includes evaluating OpenTriage with other datasets. It will be interesting to see if the monotonic trend of the gray areas is always confirmed and to study under which assumptions it can be proved. Another interesting development is to the design of a incremental solution leveraging the redundancy for finding the best source integration strategy [25] while integrating a single source at a time.

References

[1] C. Lockard, P. Shiralkar, X. L. Dong, Openceres: When open information extraction meets the semi-structured web, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 3047–3056.

[2] M. Bronzi, V. Crescenzi, P. Merialdo, P. Papotti, Extraction and integration of partially overlapping web sources, Proceedings of the VLDB Endowment 6 (2013) 805–816.
[3] Z. Bellahsene, A. Bonifati, F. Duchateau, Y. Velegrakis, On evaluating schema matching and mapping, in: Schema matching and mapping, Springer, 2011, pp. 253–291.
[4] J. Madhavan, P. A. Bernstein, E. Rahm, Generic schema matching with cupid, in: vldb, volume 1, Citeseer, 2001, pp. 49–58.
[5] E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching, the VLDB Journal 10 (2001) 334–350.
[6] D. Qiu, L. Barbosa, V. Crescenzi, P. Merialdo, D. Srivastava, Big data linkage for product specification pages, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 67–81.
[7] A. Gruenheid, X. L. Dong, D. Srivastava, Incremental record linkage, Proceedings of the VLDB Endowment 7 (2014) 697–708.
[8] S. E. Whang, H. Garcia-Molina, Incremental entity resolution on rules and data, The VLDB journal 23 (2014) 77–102.
[9] D. G. Brizan, A. U. Tansel, A. survey of entity resolution and record linkage methodologies, Communications of the IIMA 6 (2006) 5.
[10] X. L. Dong, D. Srivastava, Big data integration, in: 2013 IEEE 29th international conference on data engineering (ICDE), IEEE, 2013, pp. 1245–1248.
[11] N. Dalvi, A. Machanavajjhala, B. Pang, An analysis of structured data on the web, arXiv preprint arXiv:1203.6406 (2012).
[12] V. Cetorelli, P. Atzeni, V. Crescenzi, F. Milicchio, The smallest extraction problem, Proceedings of the VLDB Endowment 14 (2021) 2445–2458.
[13] V. Crescenzi, P. Merialdo, Wrapper inference for ambiguous web pages, Applied Artificial Intelligence 22 (2008) 21–52.
[14] V. Cetorelli, V. Crescenzi, P. Merialdo, R. Voyat, Noah: Creating data integration pipelines over continuously extracted web data., in: EDBT/ICDT Workshops, 2021.
[15] G. Papadakis, D. Skoutas, E. Thanos, T. Palpanas, A survey of blocking and filtering techniques for entity resolution, arXiv preprint arXiv:1905.06167 (2019).
[16] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, N. Tang, Distributed representations of tuples for entity resolution, Proceedings of the VLDB Endowment 11 (2018) 1454–1467.
[17] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: Vldb, volume 99, 1999, pp. 518–529.
[18] V. Christophides, V. Efthymiou, K. Stefanidis, Entity resolution in the web of data, Synthesis Lectures on the Semantic Web 5 (2015) 1–122.
[19] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, T. Palpanas, Parallel meta-blocking for scaling entity resolution over big heterogeneous data, Information Systems 65 (2017) 137–157.
[20] V. Crescenzi, A. A. Fernandes, P. Merialdo, N. W. Paton, Crowdsourcing for data management, Knowledge and Information Systems 53 (2017) 1–41.
[21] G. Li, Y. Zheng, J. Fan, J. Wang, R. Cheng, Crowdsourced data management: Overview and challenges, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1711–1716.

[22] T. S. Behrend, D. J. Sharek, A. W. Meade, E. N. Wiebe, The viability of crowdsourcing for survey research, Behavior research methods 43 (2011) 800–813.
[23] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, arXiv preprint arXiv:2004.00584 (2020).
[24] Q. Hao, R. Cai, Y. Pang, L. Zhang, From one tree to a forest: a unified solution for structured web data extraction, in: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 2011, pp. 775–784.
[25] X. L. Dong, B. Saha, D. Srivastava, Less is more: Selecting sources wisely for integration, Proceedings of the VLDB Endowment 6 (2012) 37–48.
[26] V. Crescenzi, G. Mecca, P. Merialdo, et al., Roadrunner: Towards automatic data extraction from large web sites, in: VLDB, volume 1, 2001, pp. 109–118.
[27] C. Lockard, X. L. Dong, A. Einolghozati, P. Shiralkar, Ceres: Distantly supervised relation extraction from the semi-structured web, arXiv preprint arXiv:1804.04635 (2018).
[28] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, C. Wang, Diadem: thousands of websites to a single database, Proceedings of the VLDB Endowment 7 (2014) 1845–1856.
[29] N. Bhutani, H. Jagadish, D. Radev, Nested propositions in open information extraction, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 55–64.
[30] C. Niklaus, M. Cetto, A. Freitas, S. Handschuh, A survey on open information extraction, arXiv preprint arXiv:1806.05599 (2018).
[31] H. A. Sleiman, R. Corchuelo, Trinity: on using trinary trees for unsupervised web data extraction, IEEE Transactions on Knowledge and Data Engineering 26 (2013) 1544–1556.
[32] M. A. B. M. Azir, K. B. Ahmad, Wrapper approaches for web data extraction: A review, in: 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), IEEE, 2017, pp. 1–6.
[33] M. Schmitz, S. Soderland, R. Bart, O. Etzioni, et al., Open language learning for information extraction, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 523–534.
[34] A. Fader, S. Soderland, O. Etzioni, Identifying relations for open information extraction, in: Proceedings of the 2011 conference on empirical methods in natural language processing, 2011, pp. 1535–1545.
[35] Y. Zhou, Y. Sheng, N. Vo, N. Edmonds, S. Tata, Learning transferable node representations for attribute extraction from web documents, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1479–1487.
[36] C. Lockard, P. Shiralkar, X. L. Dong, H. Hajishirzi, Zeroshotceres: Zero-shot relation extraction from semi-structured webpages, arXiv preprint arXiv:2005.07105 (2020).
[37] A. Carlson, C. Schafer, Bootstrapping information extraction from semi-structured web pages, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 195–210.
[38] B. Y. Lin, Y. Sheng, N. Vo, S. Tata, Freedom: A transferable neural architecture for structured information extraction on web documents, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1092–1102.