The VLDB Journal (2013) 22:665–687
DOI 10.1007/s00778-013-0324-z

SPECIAL ISSUE PAPER

Large-scale linked data integration using probabilistic reasoning and crowdsourcing

Gianluca Demartini · Djellel Eddine Difallah · Philippe Cudré-Mauroux

Received: 26 September 2012 / Revised: 15 June 2013 / Accepted: 20 June 2013 / Published online: 18 July 2013

Abstract We tackle the problems of semiautomatically matching linked data sets and of linking large collections of Web pages to linked data. Our system, ZenCrowd, (1) uses a three-stage blocking technique in order to obtain the best possible instance matches while minimizing both computational complexity and latency, and (2) identifies entities from natural language text using state-of-the-art techniques and automatically connects them to the linked open data cloud. First, we use structured inverted indices to quickly find potential candidate results from entities that have been indexed in our system. Our system then analyzes the candidate matches and refines them whenever deemed necessary using computationally more expensive queries on a graph database. Finally, we resort to human computation by dynamically generating crowdsourcing tasks in case the algorithmic components fail to come up with convincing results. We integrate all results from the inverted indices, from the graph database and from the crowd using a probabilistic framework in order to make sensible decisions about candidate matches and to identify unreliable human workers. In the following, we give an overview of the architecture of our system and describe in detail our novel three-stage blocking technique and our probabilistic decision framework. We also report on a series of experimental results on a standard data set, showing that our system can achieve a 95 % average accuracy on instance matching (as compared to the initial 88 % average accuracy of the purely automatic baseline) while drastically limiting the amount of work performed by the crowd. The experimental evaluation of our system on the entity linking task shows an average relative improvement of 14 % over our best automatic approach.

Keywords Instance matching · Entity linking · Data integration · Crowdsourcing · Probabilistic reasoning

1 Introduction

Semistructured data are becoming more prominent on the Web as more and more data are either interweaved or serialized in HTML pages. The linked open data (LOD) community,[1] for instance, is bringing structured data to the Web by publishing data sets using the RDF formalism and by interlinking pieces of data coming from heterogeneous sources. As the LOD movement gains momentum, linking traditional Web content to the LOD cloud is giving rise to new possibilities for online information processing. For instance, identifying unique real-world objects, persons, or concepts, in textual content and linking them to their LOD counterparts (also referred to as *Entities*), opens the door to automated text enrichment (e.g., by providing additional information coming from the LOD cloud on entities appearing in the HTML text), as well as streamlined information retrieval and integration (e.g., by using links to retrieve all text articles related to a given concept from the LOD cloud).

As more LOD data sets are being published on the Web, unique entities are getting described multiple times by different sources. It is therefore critical that such openly available

This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459.

G. Demartini (✉) · D. E. Difallah · P. Cudré-Mauroux
eXascale Infolab, University of Fribourg, Fribourg, Switzerland
e-mail: gianluca.demartini@unifr.ch

D. E. Difallah
e-mail: djelleleddine.difallah@unifr.ch

P. Cudré-Mauroux
e-mail: philippe.cudre-mauroux@unifr.ch

[1] http://linkeddata.org/.

data sets are interlinked to each other in order promote global data interoperability. The interlinking of data sets describing similar entities enables Web developers to cope with the rapid growth of LOD data, by focusing on a small set of well-known data sets (such as DBPedia[2] or Freebase[3]) and by automatically following links from those data sets to retrieve additional information whenever necessary.

Automatizing the process of *matching instances* from heterogeneous LOD data sets and the process of *linking entities* appearing in HTML pages to their correct LOD counterpart is currently drawing a lot of attention (see the Sect. 2 below). These processes represent however a highly challenging task, as instance matching is known to be extremely difficult even in relatively simple contexts. Some of the challenges that arise in this context are (1) to identify entities appearing in natural text, (2) to cope with the large-scale and distributed nature of LOD, (3) to disambiguate candidate concepts, and finally (4) to match instances across data sets.

This paper describes ZenCrowd, a system we have developed in order to create links across large data sets containing similar instances and to semiautomatically identify LOD entities from textual content. In a recent work [17], we focused on the entity linking task, that is, on extracting and identifying occurrences of LOD instances from textual content (e.g., news articles in HTML format). In the present work, we extend ZenCrowd to handle both instance matching and entity linking. Our system gracefully combines algorithmic and manual integration, by first taking advantage of automated data integration techniques and then by improving the automatic results by involving human workers.

The ZenCrowd approach addresses the scalability issues of data integration by proposing a novel three-stage blocking technique that incrementally combines three very different approaches together. In a first step, we use an inverted index built over the entire data set to efficiently determine potential candidates and to obtain an initial ranked list of potential results. Top potential candidates are then analyzed further by taking advantage of a more accurate (but also more costly) graph-based instance matching techniques (a similar structured/unstructured hybrid approach has been taken in [45]). Finally, results yielding low confidence values (as determined by probabilistic inference) are used to dynamically create *micro*-tasks published on a crowdsourcing platform, the assumption being that tasks in question do not need special expertise to be performed.

ZenCrowd does *not* focus on the algorithmic problems of instance matching and entity linking per se. However, we make a number of key contributions at the interface of algorithmic and manual data integration and discuss in detail how to most effectively and efficiently combine scalable inverted indices, structured graph queries and human computation in order to match large LOD data sets. The contributions of this paper include the following:

- a new system architecture supporting algorithmic and manual instance matching as well as entity linking in concert.
- a new three-stage blocking approach that combines highly scalable automatic filtering of semistructured data together with more complex graph-based matching and high-quality manual matching performed by the crowd.
- a new probabilistic inference framework to dynamically assess the results of arbitrary human workers operating on a crowdsourcing platform and to effectively combine their (conflicting) output taking into account the results of the automatic stage output.
- an empirical evaluation of our system in a real deployment over different Human Intelligence Task interfaces showing that ZenCrowd combines the best of both worlds, in the sense that our combined approach turns out to be more effective than both (a) pure algorithmic, by improving the accuracy and (b) full manual matching, by being cost-effective while mitigating the workers' uncertainty.

The rest of this paper is structured as follows: We review the state of the art in instance matching, entity linking, and crowdsourcing systems in Sect. 2. Section 3 introduces the terminology used throughout the paper. Section 4 gives an overview of the architecture of our system, including its algorithmic matching interface, its probabilistic inference engine, and its templating and crowdsourcing components. Section 5 presents our graph-based matching confidence measure as well as different methods to crowdsource instance matching and entity linking tasks. We describe our formal model to combine both algorithmic and crowdsourcing results using probabilistic networks in Sect. 6. We introduce our evaluation methodology and discuss results from a real deployment of our system for the instance matching task in Sect. 7 and for the entity linking task in Sect. 8, before concluding in Sect. 9.

2 Related work

2.1 Instance matching

The first task addressed by this paper is that of matching instances of multiple types among two data sets. Thanks to the LOD movement, many data sets describing instances have been created and published on the Web.

A lot of attention has been put on the task of automatic instance matching, which is defined as the identification of the same real-world object described in two different data

[2] http://www.dpbedia.org.

[3] http://freebase.org.

sets. Classical matching approaches are based on string similarities ("Barack Obama" vs. "B. Obama") such as the edit distance [33], the Jaro similarity [27], or the Jaro-Winkler similarity [50]. More advanced techniques, such as instance group linkage [40], compare groups of records to find matches. A third class of approaches uses semantic information. Reference reconciliation [21], for example, builds a dependency graph and exploits relations to propagate information among entities. Recently, approaches exploiting Wikipedia as background corpus have been proposed as well [9,13]. In [26], the authors propose entity disambiguation techniques using relations between entities in Wikipedia and concepts. The technique uses, for example, the link between "Micheal Jordan" and the "University of California, Berkeley" or to "basketball" on Wikipedia.

The number of candidate matching pairs between two data sets grows rapidly (i.e., quadratically) with the size of the data, making the matching task rapidly intractable in practice. Methods based on blocking [41,49] have been proposed to tackle scalability issues. The idea is to adopt a computationally inexpensive method to first group together candidate matching pairs and, as a second step, to adopt a more accurate and expensive measure to compare all possible pairs within the candidate set.

Crowdsourcing techniques have already been leveraged for instance matching. In [48], the authors propose a hybrid human–machine approach that exploits both the scalability of automatic methods and the accuracy of manual matching. The focus of their work is on how to best present the matching task to the crowd. Instead, our work focuses on how to combine automated and manual matching by means of a three-stage blocking technique and a probabilistic network able to identify and weight-out low-quality answers.

In idMesh [15], we built disambiguation graphs based on the transitive closures of equivalence links for networks containing uncertain information. Our present work focuses on hybrid matching techniques for LOD data sets, combining both automated processes and human computation in order to obtain a system that is both scalable and highly accurate.

2.2 Entity linking

The other task performed by ZenCrowd is entity linking, that is, identifying instances from textual content and linking them to their description in a database. Entities, that is, real-world objects described following a given schema/ontology, have recently become first-class citizens on the Web. A large amount of online search queries are about entities [42], and search engines exploit entities and structured data to build their result pages [25]. In the field of information retrieval (IR), a lot of attention has been given to entities: At TREC,[4] the main IR evaluation initiative, the task of Expert Finding, Related Entity Finding, and Entity List Completion have been studied [2,3]. Along similar lines, we have evaluated entity ranking in Wikipedia at INEX[5] recently [18].

The problem of assigning identifiers to instances mentioned in textual content (i.e., entity linking) has been widely studied by the database and the semantic Web research communities. A related effort has, for example, been carried out in the context of the OKKAM project,[6] which suggested the idea of an entity name system (ENS) to assign identifiers to entities on the Web [8]. The ENS could integrate techniques from our paper to improve matching effectiveness.

The first step in entity linking consists in extracting entities from textual content. Several approaches developed within the NLP field provide high-quality entity extraction for persons, locations, and organizations [4,12]. State-of-the-art techniques are implemented in tools like Gate [16], the Stanford Parser [30] (which we use in our experiments), and Extractiv.[7]

Once entities are extracted, they still need to be disambiguated and matched to semantically similar but syntactically different occurrences of the same real-world object (e.g., "Mr. Obama" and "President of the USA").

The final step in entity linking is that of deciding which links to retain in order to enrich the entity. Systems performing such a task are available as well (e.g., Open Calais,[8] DBPedia Spotlight [37]). Relevant approaches aim for instance at enriching documents by automatically creating links to Wikipedia pages [38,44], which can be seen as entity identifiers. While previous work selects uniform resource identifiers (URIs) from a specific corpus (e.g., DBPedia, Wikipedia), our goal in ZenCrowd is to assign entity identifiers from the larger LOD cloud[9] instead.

The present work aims at correctly linking isolated entities to external entities using an effective combination of algorithmic and manual matching techniques. To the best of our knowledge, this paper is the first to propose a principled approach based on crowdsourcing techniques to improve the quality of automated entity linking algorithms.

2.3 Ad hoc object retrieval

Another task related to entity linking is ad hoc object retrieval (AOR) [42], where systems need to retrieve the correct URIs given a keyword query representing an entity. Such a task has been evaluated in the context of the Semantic Search work-

[4] http://trec.nist.gov.

[5] https://inex.mmci.uni-saarland.de/.

[6] http://www.okkam.org.

[7] http://extractiv.com/.

[8] http://www.opencalais.com/.

[9] http://linkeddata.org/.

shop in 2010[10] and 2011[11] using a set of queries extracted from a commercial search engine query log and crowdsourcing techniques to create the gold standard. Most of the proposed systems for this task (see, for example, Blanco et al. [7]) exploit IR indexing and ranking techniques over the RDF data set used at the Billion Triple Challenge[12] 2009. Similarly to such tasks, our data set is composed of a large set of triples coming from LOD data sets, while our queries consist of instance labels from the testset where the gold standard is manually created by experts. In addition to those efforts, we selectively exploit the crowd to improve the accuracy of the task.

ZenCrowd adopts a hybrid architecture that combines unstructured inverted indices together with a structured graph database to optimize the task of instance matching. A similar approach has been taken in our previous work [45] where we combined structured and unstructured representations of graph data to effectively address the task of ad hoc object retrieval.

2.4 Crowdsourcing

ZenCrowd selectively adopts crowdsourcing to improve the quality in data integration tasks. Crowdsourcing is a term used to define those methods to generate or process data asking to a large group of people to complete small tasks. It is possible to categorize different crowdsourcing strategies based on the different types of incentives used to motivate the crowd to perform such tasks. One of the most successful example of crowdsourcing is the creation of Wikipedia, an online encyclopedia collaboratively written by a large number of web users. The incentive to create articles in Wikipedia is to help the community and to share knowledge with others.

An incentive that is often leveraged to get input from the crowd is *fun*. Games with a purpose have studied how to design entertaining applications that can generate useful data to be processed by further algorithms. An example of a successful game that at the same time generates meaningful data is the ESP game [46] where two human players have to agree on the words used to tag a picture. An extension of this game is Peekaboom: a game that asks the player to detect and annotate specific objects within an image [47].

A different type of crowdsourcing uses a monetary incentive to motivate the crowd to perform some tasks. The most popular paid crowdsourcing platform currently available is Amazon MTurk[13] where micro-tasks (called Human Intelligence Tasks or HITs) are published by requesters and selected by workers who perform them in exchange of a small monetary reward. We use the MTurk platform as a basis for the ZenCrowd system. Other paid crowdsourcing platforms use the approach of modeling worker skills to select the right worker for a specific HIT [20]. This is beneficial when the tasks are domain-specific and require workers having some domain knowledge. In this paper, we use MTurk as a crowdsourcing platform as we deal with well-known general-domain entities. Alternative platforms could be used for domain-specific data integration tasks like, for example, linking entities described in scientific articles. ZenCrowd uses paid crowdsourcing to enable fast scalability to large amounts of data. This is possible thanks to the continuous availability of human workers on crowdsourcing platforms such as Amazon MTurk.

Paid crowdsourcing is a relatively recent technique that is currently being investigated in a number of contexts. In the IR community, crowdsourcing techniques have been mainly used to create test collections for repeatable relevance assessment [1,28,29]. The task of the workers is to judge the relevance of a document for a given query. Studies have shown that this is a practically relevant approach, which produces reliable evaluation collections [6]. The database community is currently evaluating how crowdsourcing methods can be used to build RDMS systems able to answer complex queries where subjective comparison is needed (e.g., "10 papers with the most novel ideas") [22,43]. Crowdsourcing can also be used for basic computational operations such as sort and join [36] as well as for sentiment analysis and image tagging [35].

In the context of entity identification, crowdsourcing has been used by Finn et al. [23] to annotate entities in Twitter. Their goal is simpler than ours as they ask human workers to identify entities in text and assign a type (i.e., person, location, or organization) to the identified entities. Our goal is, instead, to assign entity identifiers to large numbers of entities on the Web. The two approaches might be combined to obtain high-quality results for both extraction and linking.

3 Preliminaries

As already mentioned, ZenCrowd addresses two distinct data integration tasks related to the general problem of entity resolution [24].

We define *Instance Matching* as the task of identifying two instances following different schemas (or ontologies) but referring to the same real-world object. Within the database literature, this task is related to record linkage [11], duplicate detection [5], or entity identification [34] when performed over two relational databases. However, in our setting, the main goal is to create new cross-data set $<$ `owl : sameAs` $>$ RDF statements. As commonly assumed for record linkage, we also assume that there are no duplicate entities within the same source and leverage this assumption when computing

[10] http://km.aifb.kit.edu/ws/semsearch10/.

[11] https://km.aifb.kit.edu/ws/semsearch11/.

[12] http://challenge.semanticweb.org/.

[13] http://www.mturk.com.

the final probability of a match in our probabilistic reasoning step.

We define *Entity Linking* as the task of assigning a URI selected from a background knowledge base for an entity mentioned in a textual document. This task is also known as entity resolution [24] or disambiguation [10] in the literature. In addition to the classic entity resolution task, the objective of our task is not only to understand which possible interpretation of the entity is correct (Michael Jordan the basketball player as compared to the UC Berkeley professor), but also to assign a URI to the entity, which can be used to retrieve additional factual information about it.

Given two LOD data set $U_1 = \{u_{11}, .., u_{1n}\}$ and $U_2 = \{u_{21}, .., u_{2m}\}$ containing structured entity descriptions u_{ij}, where i identifies the data set and j the entity URI, we define instance matching as the identification of each pair (u_{1i}, u_{2j}) of entity URIs from U_1 and U_2 referring to the same real-world entity and call such a pair a *match*. An example of *match* is given by the pair u_{11} = <http://dbpedia.org/resource/Tom_Cruise> and u_{21} = <http://www.freebase.com/m/07r1h> where U_1 is the DBPedia LOD data set and U_2 is the Freebase LOD data set.

Given a document d and a LOD data set $U_1 = \{u_{11}, .., u_{1n}\}$, we define entity linking as the task of identifying all entities in U_1 from d and of associating the corresponding identifier u_{1i} to each entity.

These two tasks are highly related: Instance matching aims at creating connections between different LOD data sets that describe the same real-world entity using different vocabularies. Such connections can then be used to run linking on textual documents. Indeed, ZenCrowd uses existing $<\texttt{owl}:\texttt{sameAs}>$ statements as probabilistic priors to take a final decision about which links to select for an entity appearing in a textual document.

Hence, we use in the following the term *entity* to refer to a real-world object mentioned in a textual document (e.g., a news article), while we use the term *instance* to refer to its structured description (e.g., a set of RDF triples), which follows the well-defined schema of a LOD data set.

Our system relies on LOD data sets for both tasks. Such linked data sets describe interconnected entities that are commonly mentioned in Web content. As compared to traditional data integration tasks, the use of LOD data may support integration algorithms by means of its structured entity descriptions and entity interlinking within and across data sets (Fig. 1).

In our work, we make use of Human Intelligence at scale to, first, improve the quality of such links across data sets and, second, to connect unstructured documents to the structured representation of the entities they mention. To improve the result for both tasks, we selectively use paid micro-task crowdsourcing. To do this, we create HITs on a crowdsourcing platform. For the entity linking task, a HIT consists of asking which of the candidate links is correct for an entity extracted from a document. For the instance matching task, a HIT consists in finding which instance from a target data set corresponds to a given instance from a source data set. See Figs. 2, 3, and 4, which give examples of such tasks.

Paid crowdsourcing presents enormous advantages for high-quality data processing. The disadvantages, however, potentially include the following: high financial cost, low availability of workers, and poor workers' skills or honesty. To overcome those shortcomings, we alleviate the financial cost using an efficient decision engine that selectively picks tasks that have a high improvement potential. Our present assumption is that entities extracted from HTML news articles could be recognized by the large public, especially when provided with sufficient contextual information. Furthermore, each task is shown to multiple workers to balance out low-quality answers.

4 Architecture

ZenCrowd is a hybrid platform that takes advantage of both algorithmic and manual data integration techniques simultaneously. Figure 1 presents a simplified architecture of our system. We start by giving an overview of our system below in Sect. 4.1 and then describe in more detail some of its components in Sects. 4.2–4.4.

4.1 System overview

In the following, we describe the different components of the ZenCrowd system focusing first on the instance matching and then on the entity linking pipeline.

4.1.1 Instance matching pipeline

In order to create new links, ZenCrowd takes as input a pair of data sets from the LOD cloud. Among the two data sets, one is selected as the *source* data set and one as the *target* data set. Then, for each instance of the source data set, our system tries to come up with candidate matches from the target data set.

First, the label used to name the source instance is used to query the *LOD Index* (see Sect. 4.2) in order to obtain a ranked list of candidate matches from the target data set. This can efficiently, and cheaply, filter out numerous *clear* non-matches out of potentially numerous (in the order of hundreds of millions for some LOD data sets) instances available. Next, top-ranked candidate instances are further examined in the graph database. This step is taken to obtain more complete information about the target instances, both to compute a more accurate matching score and to provide information to the *Micro-Task Manager* (see Fig. 1), which has to fill the

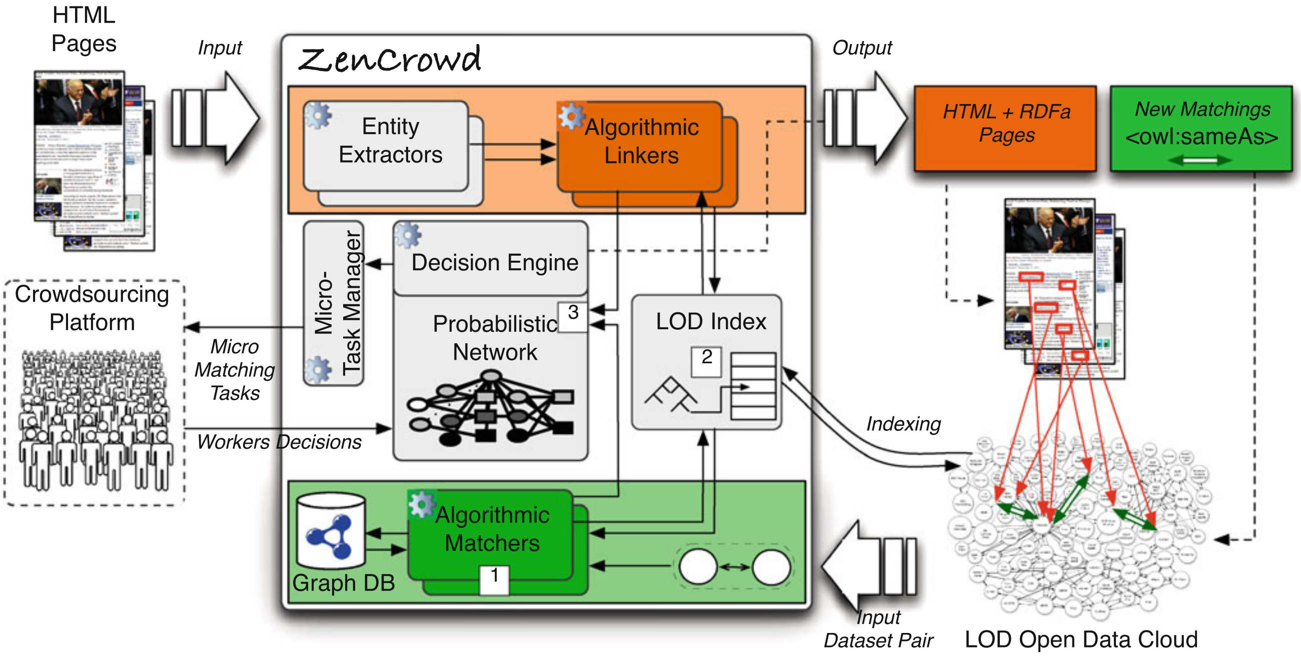
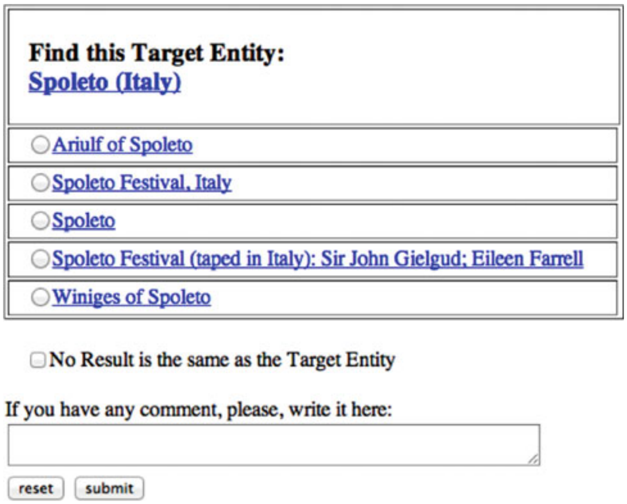


Fig. 1 The architecture of ZenCrowd: For the instance matching task (*green pipeline*), the system takes as input a pair of data sets to be interlinked and creates new links between the data sets using < `owl` : `sameAs` > RDF triples. ZenCrowd uses a three-stage blocking procedure that combines both algorithmic matchers and human workers in order to generate high-quality results. For the entity linking task (*orange pipeline*), our system takes as input a collection of HTML pages and enriches them by extracting textual entities appearing in the pages and linking them to the linked open data cloud (color figure online)



Find this Target Entity:
Spoleto (Italy)

○ Ariulf of Spoleto

○ Spoleto Festival, Italy

○ Spoleto

○ Spoleto Festival (taped in Italy): Sir John Gielgud; Eileen Farrell

○ Winiges of Spoleto

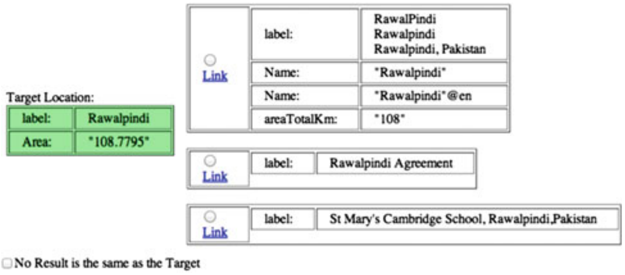
□ No Result is the same as the Target Entity

If you have any comment, please, write it here:

Fig. 2 *Label-only* instance matching HIT interface, where entities are displayed as textual labels linking to the full entity descriptions in the LOD cloud

HIT templates for the crowd (see Sect. 4.5, which describes our three-stage blocking methodology in more detail).

At this point, the candidate matches that have a low confidence score are sent to the crowd for further analysis. The *Decision Engine* collects confidence scores from the previous steps in oder to decide what to crowdsource, together with data from the graph database to construct the HITs.



Target Location:

label:	Rawalpindi
Area:	"108.7795"

○ Link

label:	RawalPindi Rawalpindi Rawalpindi, Pakistan
Name:	"Rawalpindi"
Name:	"Rawalpindi"@en
areaTotalKm:	"108"

○ Link

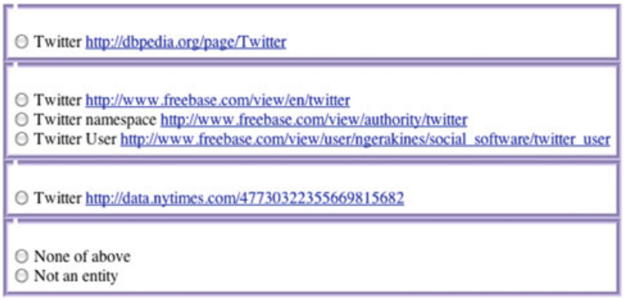
label:	Rawalpindi Agreement

○ Link

label:	St Mary's Cambridge School, Rawalpindi,Pakistan

□ No Result is the same as the Target

Fig. 3 *Molecule* instance matching HIT interface, where the labels of the entities as well as related property-value pairs are displayed



○ Twitter http://dbpedia.org/page/Twitter

○ Twitter http://www.freebase.com/view/en/twitter
○ Twitter namespace http://www.freebase.com/view/authority/twitter
○ Twitter User http://www.freebase.com/view/user/ngerakines/social_software/twitter_user

○ Twitter http://data.nytimes.com/47730322355669815682

○ None of above
○ Not an entity

Fig. 4 Entity linking HIT interface

Finally, we gather the results provided by the crowd into to the *Probabilistic Network* component, which combines them to come up with a final matching decision. The generated matchings are then given as output by ZenCrowd in the form of RDF $<$ `owl` : `sameAs` $>$ links that can be added back to the LOD cloud.

4.1.2 Entity linking pipeline

The other task that ZenCrowd performs is entity linking, that is, identifying occurrences of LOD entities in textual content and creating links from the text to corresponding instances stored in a database. ZenCrowd takes as input sets of HTML pages (that can, for example, be provided by a Web crawler). The HTML pages are then passed to *Entity Extractors* that inspect the pages and identify potentially relevant textual entities (e.g., persons, companies, places, etc.) mentioned in the page. Once detected, the entities are fed into *Algorithmic Linkers* that attempt to automatically link the textual entities to semantically similar instances from the LOD cloud. As querying the Web of data dynamically to link each entity would incur a very high latency, we build a local cache (called *LOD Index* in Fig. 1) to locally retrieve and index relevant information from the LOD cloud. Algorithmic linkers return lists of top-k *links* to LOD entities, along with a confidence value for each potentially relevant link.

The results of the algorithmic linkers are stored in a *Probabilistic Network* and are then combined and analyzed using probabilistic inference techniques. ZenCrowd treats the results of the algorithmic linkers in three different ways depending on their quality. If the algorithmic results are deemed excellent by our decision engine, the results (i.e., the links connecting a textual entity extracted from an HTML page to the LOD cloud) get stored in a local database directly. If the results are deemed useless (e.g., when all the links picked by the linkers have a low confidence value), the results get discarded. Finally, if the results are deemed promising but uncertain (for example, because several algorithmic linkers disagree on the links, or because their confidence values are relatively low), they are then passed to the *Micro-Task Manager*, which extracts relevant snippets from the original HTML pages, collects all promising links, and dynamically creates a micro-task using a templating engine. An example of micro-task for the entity linking pipeline is shown in Fig. 4. Once created, the micro-task is published on a crowdsourcing platform, where it is handled by collections of human workers. When the human workers have performed their task (i.e., when they have picked the relevant links for a given textual entity), workers results are fed back to the probabilistic network. When all the links are available for a given HTML page, an enriched HTML page—containing both the original HTML code and RDFa annotations linking the textual entities to their counterpart from the LOD cloud—is finally generated.

4.2 LOD index and graph database

The LOD index is a declarative information retrieval engine used to speedup the entity retrieval process. While most LOD data sets provide a public SPARQL interface, they are in practice very cumbersome to use due to the very high latency (from several hundreds of milliseconds to several seconds) and bandwidth consumption they impose. Instead of querying the LOD cloud dynamically for each new instance to be matched, ZenCrowd caches locally pertinent information from the LOD cloud. Our LOD index engine receives as input a list of SPARQL endpoints or LOD dumps as well as a list of triple patterns, and iteratively retrieves all corresponding triples from the LOD data sets. Using multiple LOD data sets improves the coverage of our system, since some data sets cover only geographical locations, while other data sets cover the scientific domain or general knowledge. The information thus extracted is cached locally in two ways: in our efficient analytical graph query engine [51]—offering a SPARQL interface—and in an inverted index to provide efficient support for unstructured queries.

After ranked results are obtained from the LOD index, a more in-depth analysis of the candidate matches is performed by means of queries to a graph database. This component stores and indexes data from the LOD data sets and accepts SPARQL queries to retrieve predicate value pairs attached to the query node. This component is used both to define the confidence scoring function by means of schema matching results (Sect. 5.1) and to compute confidence scores for candidate matches and to show matching evidence to the crowd (Sect. 5.2).

4.3 Probabilistic graph and decision engine

Instead of using heuristics or arbitrary rules, ZenCrowd systematizes the use of probabilistic networks to make sensible decisions about the potential instance matches and entity links. All evidences gathered from both the algorithmic methods and the crowd are fed into a scalable probabilistic store and used by our decision engine to process all entities accordingly. Our probabilistic models are described in detail in Sect. 6.

4.4 Extractors, algorithmic linkers, and algorithmic matchers

The extractors and algorithmic linkers are used exclusively by the entity linking pipeline (see Fig. 1). The entity extractors receive HTML as input, and extract named entities appearing in the HTML content as output. Entity extraction is an active area of research and a number of advances have recently been made in that field (using for instance third-party information or novel NLP techniques). Entity extraction is not the focus of our work in ZenCrowd. However, we support arbitrary entity extractors through a generic interface in our system and *union* their respective output to obtain additional results.

Once extracted, the textual entities are inspected by algorithmic linkers, whose role is to find semantically related entities from the LOD cloud. ZenCrowd implements a number of state-of-the-art linking techniques (see Sect. 8 for more details) that take advantage of the LOD index component to efficiently find potential matches. Each matcher also implements a normalized scoring scheme, whose results are combined by our decision engine (see Sect. 6).

4.5 Three-stage blocking for crowdsourcing optimization

For the instance matching pipeline, a naive implementation of an algorithmic matcher would check each pair of instances from two input data sets. However, the problem of having to deal with too many candidate pairs rapidly surfaces. Moreover, crowdsourcing all possible candidate pairs is unrealistic: For example, matching two data sets containing just 1,000 instances each would cost $150,000 if we crowdsource 1,000,000 possible pairs to 3 workers paying $0.05 per task. Instead, we propose a three-stage blocking approach.

A common way to deal with the quadratic number of potential comparisons is *blocking* (see Sect. 2). Basically, blocking groups promising candidate pairs together in sets using a computationally inexpensive method (e.g., clustering) and, as a second step, performs all possible comparisons within such sets using a more expensive method (e.g., string similarity).

ZenCrowd uses a novel three-stage blocking approach that involves crowdsourcing as an additional step in the blocking process (see the three stages in Fig. 1). Crowdsourcing the instance matching process is expensive both in terms of latency and financially. For this reason, only a very limited set of candidate pairs should be crowdsourced when matching large data sets.

Given a source instance from a data set, ZenCrowd considers all instances of the target data set as possible matches. The first blocking step is performed by means of an inverted index over the labels of all instances in the target data set. This allows to produce a list of instances ranked by a scoring function that measures the likelihood of matching the source instance very efficiently (i.e., in the order of milliseconds).

As a second step, ZenCrowd computes a more accurate but also more computationally expensive matching confidence for the top-ranked instances generated by the first step. This confidence value is computed based on schema matching results among the two data sets and produces a score in [0, 1]. This value is not computed on all instances of the target data set but rather for those that are likely to be a good match as given by the first blocking step (see Sect. 5.1).

This hybrid approach exploiting the interdependence of unstructured indices as well as structured queries against a graph database is similar to the approach taken in our previous work [45] where, for the task of ad hoc object retrieval, a ranked list of results is improved by means of an analysis of the result vicinity in the graph.

The final step consists in asking the crowd about candidate matching pairs. Based on the confidence score computed during the previous step, ZenCrowd takes a decision about which HITs to create on the crowdsourcing platform. As the goal of the confidence score is to indicate how likely it is that a pair is a correct match, the system selects those cases where the confidence is not already high enough so that it can be further improved by asking the crowd. Possible instantiations of this step may include the provision of a fixed budget for the crowdsourcing platform, which the system is allowed to spend in order to optimize the quality of the results. Generally speaking, the system produces a ranked list of candidate pairs to be crowdsourced based on the confidence score. Then, given the available resources, top pairs are crowdsourced by batch to improve the accuracy of the matching process. On the other hand, improving the task completion time can be obtained by increasing the reward assigned to workers.

4.6 Micro-task manager

The micro-task manager is responsible for dynamically creating human computation tasks that are then published on a crowdsourcing platform. Whenever a match is deemed promising by our decision engine (see below for details), it is sent to the crowd for further examination. The micro-task manager dynamically builds a Web page to be published on the crowdsourcing platform using three resources: i) the name of the source instance, ii) some contextual information generated by querying the graph database, and iii) the current top-*k* matches for the instance from the blocking process. Once created and published, the matching micro-tasks can be selected by workers on the crowdsourcing platform, who are then asked to select the relevant matches (if any) for the source instance, given its name, the contextual information from the graph database, and the various candidate matches described as in the LOD cloud. Once performed, the results of the micro-matching tasks are sent back to the micro-task manager, which inserts them in the probabilistic network.

5 Effective instance matching based on confidence estimation and crowdsourcing

In this section, we describe the final steps of the blocking process that assure high-quality instance matching results.

Table 1 Top-ranked schema element pairs in DBPedia and Freebase for the person, location, and organization instances

DBPedia	Freebase
Organization	
http://www.w3.org/2000/01/rdf-schema#label	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/property/established	http://rdf.freebase.com/ns/education.educational_institution.founded
http://dbpedia.org/property/foundation	http://rdf.freebase.com/ns/business.company.founded
http://dbpedia.org/property/companyName	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/property/founded	http://rdf.freebase.com/ns/sports.sports_team.founded
Person	
http://www.w3.org/2000/01/rdf-schema#label	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/ontology/birthdate	http://rdf.freebase.com/ns/people.person.date_of_birth
http://dbpedia.org/property/name	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/property/dateOfBirth	http://rdf.freebase.com/ns/people.person.date_of_birth
http://dbpedia.org/property/dateOfDeath	http://rdf.freebase.com/ns/people.deceased_person.date_of_death
http://dbpedia.org/property/birthname	http://rdf.freebase.com/ns/common.topic.alias
Location	
http://www.w3.org/2000/01/rdf-schema#label	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/property/establishedDate	http://rdf.freebase.com/ns/location.dated_location.date_founded
http://dbpedia.org/ontology/demonym	http://rdf.freebase.com/ns/freebase.linguistic_hint.adjectival_form
http://dbpedia.org/property/name	http://rdf.freebase.com/ns/type.object.name
http://dbpedia.org/property/isocode	http://rdf.freebase.com/ns/location.administrative_division.iso_3166_2_code
http://dbpedia.org/property/areaTotalKm	http://rdf.freebase.com/ns/location.location.area

We first define our schema-based matching confidence measure, which is then used to decide which candidate matchings to crowdsource. Then, we present different approaches to crowdsourcing instance matching tasks. Specifically, we compare two different HIT designs where different context information about the instances is presented to the worker.

5.1 Instance-based schema matching

While using the crowd to match instances across two data sets typically results in high-quality matchings, it is often infeasible to crowdsource all potential matches because of the very high financial cost associated. Thus, as a second filtering step, we define a new measure that computes the confidence of a matching as generated by the initial inverted index blocking step.

Formally, given a candidate matching pair $(i1, i2)$, we define a function $f(i1, i2)$ that creates a ranked list of candidate pairs such that the pairs ranked at the top are the most likely to be correct. In such a way, it is possible to selectively crowdsource candidate matchings with lower confidence to improve matching precision with a limited cost.

The matching confidence measure used by ZenCrowd is based on schema matching information. The first step in the definition of the confidence measure consists in using a training set of matchings among the two data sets.[14] Given a training pair $(t1, t2)$, we retrieve all predicates and values for the instances $t1$ and $t2$ and perform an exact string match comparison of their values. At the end of such process, we rank predicate pairs by the number of times an exact match on their values has occurred. Table 1 gives the top-ranked predicate pairs for the DBPedia and Freebase data sets. We observe that this simple instance-based schema mapping technique yields excellent results for many LOD schemas, for instance, for the entity-type person in Table 1, where ‘birthdate’ from DBPedia is correctly matched to ‘date_of_birth’ from Freebase.

After the list of schema elements have been matched across the two data sets, we define the confidence measure for an individual candidate matching pair. To obtain a confidence score in [0, 1], we compute the average Jaccard similarity among all tokenized values of all matched schema elements for the two candidate instances u1 and u2. In the case where a list of values is assigned to a schema element (e.g., a DBPedia instance may have multiple labels that represent the instance name in different languages), we retain the maximum Jaccard similarity value in the list for that schema element. For example, the confidence score of the following matching pairs is as follows:

[14] In our experiments, we use 100 ground truth matchings that are discarded later when evaluating the proposed matching approaches.

u1	u2
`rdfs:label` barack h. obama	`fb:name` barack obama
`dbp:dateOfBirth` 08-04-61	`fb:date_of_birth` 08-04-61

$\frac{(2/3)+(1)}{2} = 0.83.$

5.2 Instance matching with the crowd

We now turn to the description of two HIT designs we experimented with for crowdsourcing instance matching in ZenCrowd. Previous work also compared different interfaces to crowdsourcing instance matching tasks [48]. Specifically, the authors compared pairwise and table-based matching interfaces. Instead, we compare matching interfaces based on different pieces of information given to the worker directly on the HIT page.

Figures 2 and 3 show our two different interfaces for the instance matching task. The *label-only* matching interface asks the crowd to find a target entity among the proposed matches. In this case, the target entity is presented as its label with a link to the corresponding LOD webpage. Then, the top-ranked instances from the DBPedia data set, which are candidates to match the target entity, are shown. This interface is reminiscent of the automatic approach based on the inverted index that performs the initial blocking step though on a larger scale (i.e., only few candidates are shown to the worker in this case).

The *molecule* interface also asks the worker to identify the target entity (from Freebase in the figure) in the table containing top-ranked entities from DBPedia. This second interface defines a simpler task for the worker by presenting directly on the HIT page relevant information about the target entity as well as about the candidate matches. In this second version of the interface, the worker is asked to directly match the instance on the left with the corresponding instance on the right. Compared to the first matching interface, the *molecule* interface does not just display the labels but also additional information (property and value pairs) about each instance. Such information is retrieved from the graph database and displayed to the worker.

In both interfaces, the worker can select the “No match” option if no instance matches the target entity. An additional field is available for the worker to leave comments.

6 Probabilistic models

ZenCrowd exploits probabilistic models to make sensible decisions about candidate results. We describe below the probabilistic models used to systematically represent and combine information in ZenCrowd, and how those models are implemented and handled by our system. We start by giving an overview of probabilistic networks first.

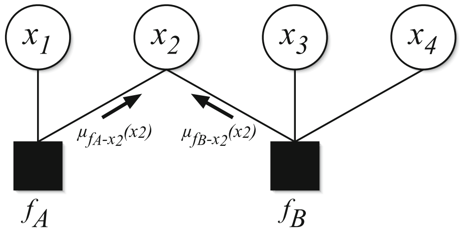


Fig. 5 A simple factor graph of four variables and two factors

6.1 A quick reminder on factor graphs and message-passing schemes

We use factor graphs to graphically represent probabilistic variables and distributions in the following. Note that our approach is not bound to this representation—we could use series of conditional probabilities only or other probabilistic graphical model—but we decided to use factor graphs for their illustrative merits.

We give below a brief introduction to factor graphs and message-passing techniques. For a more in-depth coverage, we refer the interested reader to one of the many overviews on this domain, such as Kschischang et al. [32]. Probabilistic graphical models are a marriage between probability theory and graph theory. In many situations, one can deal with a complicated global problem by viewing it as a factorization of several local functions, each depending on a subset of the variables appearing in the global problem. As an example, suppose that a global function $g(x_1, x_2, x_3, x_4)$ factors into a product of two local functions f_A and f_B: $g(x_1, x_2, x_3, x_4) = f_A(x_1, x_2) f_B(x_2, x_3, x_4)$. This factorization can be represented in a graphical form by the *factor-graph* depicted in Fig. 5, where variables (circles) are linked to their respective factors (black squares).

Often, one is interested in computing a *marginal* of this global function, e.g.,

$$\begin{aligned} g_2(x_2) &= \sum_{x_1} \sum_{x_3} \sum_{x_4} g(x_1, x_2, x_3, x_4) \qquad (1) \\ &= \sum_{\sim\{x_2\}} g(x_1, x_2, x_3, x_4) \end{aligned}$$

where we introduce the summary operator $\sum_{\sim\{x_i\}}$ to sum over all variables but x_i. Such marginals can be derived in an efficient way by a series of simple *sum-product* operations on the local function, such as

$$g_2(x_2) = \left(\sum_{x_1} f_A(x_1, x_2) \right) \left(\sum_{x_3} \sum_{x_4} f_B(x_2, x_3, x_4) \right) \quad (2)$$

Interestingly, the above computation can be seen as the product of two messages $\mu_{f_A \to x_2}(x_2)$ and $\mu_{f_B \to x_2}(x_2)$ sent, respectively, by f_A and f_B to x_2 (see Fig. 5). The *sum-product* algorithm [32] exploits this observation to compute all marginal functions of a factor graph in a concurrent and efficient manner.

Message-passing algorithms traditionally compute marginals by sending two messages — one in each direction — for every edge in the factor graph:

variable x to local factor f:

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \backslash \{f\}} \mu_{h \to x}(x) \quad (3)$$

local factor f to variable x

$$\mu_{f \to x}(x) = \sum_{\sim \{x\}} \left(f(X) \prod_{y \in n(f) \backslash \{x\}} \mu_{y \to f}(y) \right) \quad (4)$$

where $n(\cdot)$ stands for the neighbors of a variable/function node in the graph and $X = n(f)$. These computations are known to be exact for cycle-free factor graphs; in contrast, applications of the sum-product algorithm in a factor graph with cycles only result in approximate computations for the marginals [39]. However, some of the most exciting applications of the sum-product algorithms (e.g., decoding of turbo or LDPC codes) arise precisely in such situations. We show below that this is also the case for factor graphs modeling instance matching graphs.

6.2 Graph models

We start by describing the probabilistic graphs used to combine all matching evidences gathered for a given candidate URI. Consider an instance from the source data set. The candidate matches are stored as a list of potential matchings m_j from a LOD data set. Each m_j has a prior probability distribution pm_j computed from the confidence matching function. Each candidate can also be examined by human workers w_i performing micro-matching tasks and performing *clicks* c_{ij} to express the fact that a given candidate matching corresponds (or not) to the source instance from his/her perspective.

Workers, matchings, and clicks are mapped onto binary variables in our model. Workers accept two values $\{Good, Bad\}$ indicating whether they are reliable or not. Matchings can either be $Correct$ or $Incorrect$. As for click variables, they represent whether the worker i considers that the source instance is the same as the proposed matching m_j $(Correct)$ or not $(Incorrect)$. We store *prior distributions*—which represent a priori knowledge obtained, for example, through training phases or thanks to external sources—for each workers $(pw_i())$ and each matching $(pm_j())$. The clicks are *observed variables* and are set to $Correct$ or $Incorrect$ depending on how the human workers clicked on the crowdsourcing platform.

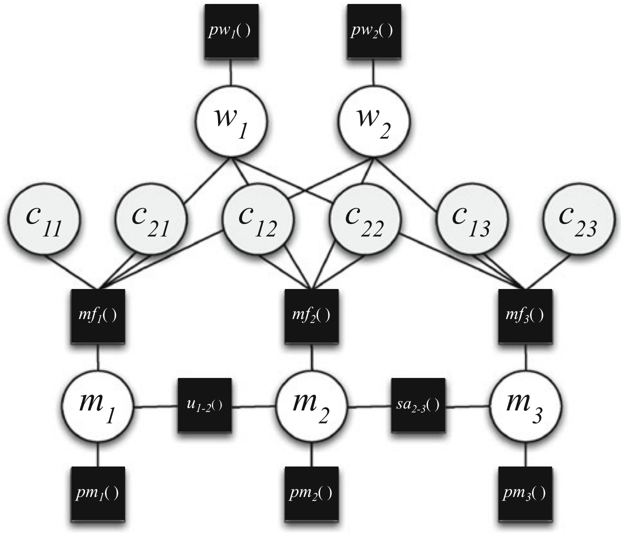


Fig. 6 Entity factor graph connecting two workers (w_i), six clicks (c_{ij}), and three candidate matchings (m_j)

A simple example of such an entity graph is given in Fig. 6. Clicks, workers, and matchings are further connected through two factors described below.

The same network can be instantiated for each entity of an entity linking task where m_j are candidate links from the LOD instead.

6.2.1 Matching and linking factors

Specific task (either matching or linking) factors $mf_j()$ connect each candidate to its related clicks and the workers who performed those clicks. Examining the relationships between those three classes of variables, we make two key observations: (1) Clicks from reliable workers should weight more than clicks from unreliable workers (actually, clicks from consistently unreliable workers deciding randomly whether a given answer is relevant or not should have no weight at all in our decision process) and (2) when reliable workers do not agree, the likelihood of the answer being correct should be proportional to the fraction of good workers indicating the answer as correct. Taking into account both observations, and mapping the value 0 to $Incorrect$ and 1 to $Correct$, we write the following function for the factor:

$$mf(w_1, \ldots, w_m, c_1, \ldots, c_n, m) \\ = \begin{cases} 0.5, \text{ if } \forall w_i \in \{w_1, \ldots, w_m\} \; w_i = Bad \\ \frac{\sum_i \mathbb{1}_{(w_i = Good \, \wedge \, c_i = m)}}{\sum_i \mathbb{1}_{(w_i = Good)}}, \text{ otherwise} \end{cases} \quad (5)$$

where $\mathbb{1}_{(cond)}$ is an indicator function equal to 1 when *cond* is true and 0 otherwise.

6.2.2 *Unicity constraints for entity linking*

Given that the instance matching task definition assumes that only one instance from the target data set can be a correct match for the source instance. Similarly, a concept appearing in textual content can only be mapped to a single entity from a given data set. We can thus rule out all configurations where more than one candidate from the same LOD data set are considered as $Correct$. The corresponding factor $u()$ is declared as being equal to 1 and is defined as follows:

$$u(m_1, \ldots, m_n) = \begin{cases} 0, \text{ if } \exists (m_i, m_j) \in \{m_1, \ldots, m_n\} \\ \quad \mid m_i = m_j = Correct \\ 1, \text{ otherwise} \end{cases} \quad (6)$$

6.2.3 *SameAs constraints for entity linking*

SameAs constraints are exclusively used in entity linking graphs. They exploit the fact that the resources identified by the links to the LOD cloud can themselves be interlinked (e.g., *dbpedia:Fribourg* is connected through an *owl:sameAs* link to *fbase:Fribourg* in the LOD cloud).[15] Considering that the *SameAs* links are correct, we define a constraint on the variables connected by *SameAs* links found in the LOD cloud; the factor $sa()$ connecting those variables puts a constraint forbidding assignments where the variables would not be set to the same values as follows:

$$sa(l_1, \ldots, l_n) = \begin{cases} 1 \text{ if } \quad \forall (l_i, l_j) \in \{l_1, \ldots, l_n\} \; l_i = l_j \\ 0 \text{ otherwise} \end{cases}$$

We enforce the constraint by declaring $sa() = 1$. This constraint considerably helps the decision process when strong evidences (good priors, reliable clicks) are available for any of the URIs connected to a *SameAs* link. When not all *SameAs* links should be considered as correct, further probabilistic analyses (e.g., on the transitive closures of the links as defined in idMesh [15]) can be put into place.

6.3 Reaching a decision

Given the scheme above, we can reach a sensible decision by simply running a probabilistic inference method (e.g., the sum-product algorithm described above) on the network and considering as correct all matchings with a posterior probability $P(l = Correct) > 0.5$. The decision engine can also consider a higher threshold $\tau > 0.5$ for the decisions in order to increase the precision of the results.

6.4 Updating the priors

Our computations always take into account prior factors capturing a priori information about the workers. As time passes, decisions are reached on the correctness of the various matches, and the probabilistic network iteratively accumulates posterior probabilities on the reliability of the workers. Actually, the network gets new posterior probabilities on the reliability of the workers for every new matching decision that is reached. Thus, the decision engine can decide to modify the priors of the workers by taking into account the evidences accumulated thus far in order to get more accurate results in the future. This corresponds to a learning parameters phase in a probabilistic graphical model when some of the observations are missing. Several techniques might be applied to this type of problem (e.g., Monte Carlo methods, Gaussian approximations). We use in the following a simple expectation–maximization [14,19] process, which looks as follows:

- Initialize the prior probability of the workers using a training phase during which workers are evaluated on k matches whose results are known. Initialize their prior reliability to $\#correct\_results/k$. If no information is available or no training phase is possible, start with $P(w = reliable) = P(w = unreliable) = 0.5$ (maximum entropy principle).
- Gather posterior evidences on the reliability of the workers $P(w = reliable | m_i = Correct/Incorrect)$ as soon as a decision is reached on a matching. Treat these evidences as new observations on the reliability of the workers, and update their prior beliefs iteratively as follows:

$$P(w = reliable) = \sum_{i=1}^{k} P_i(w = reliable | m_i) k^{-1} \quad (7)$$

where i runs over all evidences gathered so far (from the training phase and from the posterior evidences described above). Hence, we make the prior values slowly converge to their maximum likelihood to reflect the fact that more and more evidences are being gathered about the mappings as we reach more decisions on the instances. This technique can also be used to identify and *blacklist* unreliable workers dynamically.

[15] We can already see the benefit of having better matchings across data sets for that matter.

6.5 Selective model instantiation

The framework described above actually creates a gigantic probabilistic graph, where all instances, clicks, and workers are indirectly connected through various factors. However, only a small subset of the variables needs to be considered by the inference engine at any point in time. Our system updates the various priors iteratively, but only instantiates the handful of variables useful for reaching a decision on the entity currently examined. It thus dynamically instantiates instance matching and entity linking factor graphs, computes posterior probabilities for the matchings and linking, reaches a decision, updates the priors, and stores back all results before de-instantiating the graph and moving to the next instance/entity.

7 Experiments on instance matching

In this section, we experimentally evaluate the effective of ZenCrowd for the instance matching (IM) task. ZenCrowd is a relatively sophisticated system involving many components. In the following, we present and discuss the results of a series of focused experiments, each designed to illustrate the performance of a particular feature of our IM pipeline. We present extensive experimental results evaluating the entity linking pipeline (depicted using an orange background in Fig. 1) in Sect. 8. Though many other experiments could have been performed, we believe that the set of experiments presented below gives a particularly accurate account of the performance of ZenCrowd for the IM task. We start by describing our experimental setting below.

7.1 Experimental setting

To evaluate the ZenCrowd IM pipeline based on probabilistic networks as well as on crowdsourcing, we use the following data sets: The ground truth matching data come from the data interlinking task from the instance matching track of the ontology alignment evaluation initiative (OAEI) in 2011.[16] In this competition, the task was to match a given New York Times (NYT) URI[17] to the corresponding URI in DBPedia, Freebase, and Geonames. The evaluation of automatic systems is based on manual matchings created by the NYT editorial team. Starting from such data, we obtained the corresponding Freebase-to-DBPedia links via transitivity through NYT instances. Thus, the ground truth is available for the task of matching a Freebase instance to the corresponding one in DBPedia, which is more challenging than the original task as both Freebase and DBPedia are very large data sets generated semiautomatically as compared to NYT data, which is small and manually curated.

In addition, we use a standard graph data set containing data about all instances in our testset (that is, the Billion Triple Challenge BTC 2009 data set[18]) in order to run our graph-based schema matching approach and to retrieve data that is presented to the crowd. The BTC 2009 consists of a crawl of RDF data from the Web containing more than one billion facts about 800 million instances.

First blocking phase: LOD indexing and instance ranking. In order to select candidate matchings for the source instance, we adopt IR techniques similar to those that have been used by participants of the entity search evaluation at the Semantic Search workshop for the AOR task, where a string representing an entity (i.e., the query) is used to rank URIs that identify the entity. We build an inverted index over 40 million instance labels in the considered LOD data sets and run queries against it using the source instance labels in our test collection. Unless specified otherwise, the top-5 results ranked by TF-IDF are used as candidates for the crowdsourcing task after their confidence score has been computed.

Micro-task generation and ZenCrowd aggregation. To evaluate the quality of each step in the ZenCrowd IM pipeline, we selected a subset of 300 matching pairs from the ground truth of different categories (100 persons, 100 locations, and 100 organizations). Then, we crowdsourced the entire collection to compare the quality of the crowd matching against other automatic matching techniques and their combinations.

The crowdsourcing tasks were run over Amazon Mechanical Turk[19] as two independent experiments for the two proposed matching interfaces (see Sect. 5.2). Each matching task has been assigned to five different workers and was remunerated $0.05 each, employing a total of 91 workers.[20]

We aggregate the results from the crowd using the method described in Sect. 6, with an initial training phase consisting of 5 entities and a second, continuous training phase, consisting of 5 % of the other entities being offered to the workers (i.e., the workers are given a task whose solution is known by the system every 20 tasks on average).

Evaluation measures. In order to evaluate the effectiveness of the different components, we compare—for each instance—the selected matches against the ground truth that provides matching/non-matching data for each source instance. Specifically, we compute (P)recision and (R)ecall

[16] http://oaei.ontologymatching.org/2011/instance/.

[17] http://data.nytimes.com/.

[18] http://km.aifb.kit.edu/projects/btc-2009/.

[19] http://www.mturk.com.

[20] The testset we have created together with the matching results from the crowd is available for download at the page: http://exascale.info/ZenCrowd.

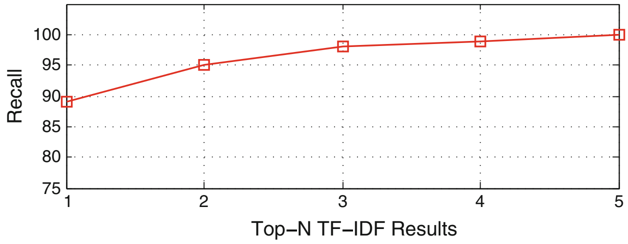


Fig. 7 Maximum achievable precision by considering top-K results from the inverted index

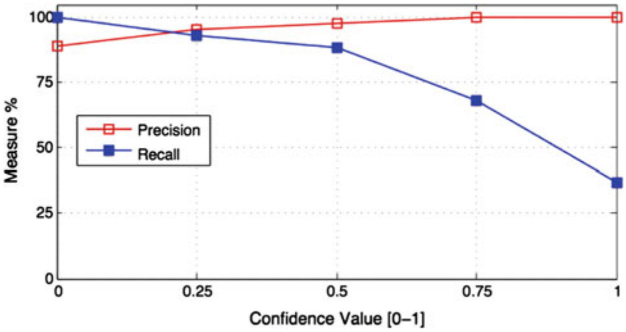


Fig. 8 Precision and recall as compared to matching confidence values

which are defined as follows: We consider as true positives (*tp*) all cases where both the ground truth and the approach select the same matches, false positives (*fp*) the cases where the approach selects a match which is not considered as correct by the ground truth, and false negatives (*fn*) the cases where the approach does non select a match, while the ground truth does. Then, precision is defined as $P = tp/(tp + fp)$ and recall as $R = tp/(tp + fn)$.

In the following, all the final matching approaches (automatic, crowd agreement vote, and ZenCrowd) are optimized to return high precision values. We decided to focus on precision from the start, since from our experience, it is the most useful metric in practice, but we have observed that high recall is obtained in most configurations.

7.2 Experimental results

In the following, we report the experimental results aiming at comparing the effectiveness of different matching techniques at different stages of the blocking process. In detail, we compare the results of our inverted index-based matching, which is highly scalable but not particularly effective, the matching based on schema information, and the matching provided by the crowd whose results are excellent but which is not *cost* and *time* efficient because of the high monetary cost it necessitates and of the high latency it generates.

Recall of the first blocking phase. The first evaluation we perform is centered on the initial blocking phase based on keyword queries over the inverted index. It is critical that such a step, while being efficiently performed over a large amount of potential candidate matchings, preserves as many correct results as possible in the generated ranked list (i.e., high recall) in order for the subsequent matching phases to be effective. This allows the graph and crowd-based matching schemes to focus on high precision in turn.

Figure 7 shows how recall varies by considering the top-N results as ranked by the inverted index using TF-IDF values. As we can see, we retrieve the correct matches for all the instances in our testset after five candidate matches already.

Second blocking phase: matching confidence function. The second blocking step involves the use of a matching confidence measure. This function measures the likelihood of a match given a pair of instances based on schema matching results and string comparison on the values directly attached to the instances in the graph (see Sect. 4.5). The goal of such a function is to be able to identify the matching pairs that are worth to crowdsource in order to improve the effectiveness of the system.

Figure 8 shows how precision and recall vary by considering matching pairs that match best according to our schema-based confidence measure. Specifically, by setting a threshold on the confidence score, we can let the system focus either on high precision or on high recall. For instance, if we only trust matches with a confidence value of 1.0, then precision is at is maximum (100 %), but the recall is low (25 %). That is, we would need to initiate many crowdsourcing tasks to compensate.

Final phase: crowdsourcing and probabilistic reasoning. After the confidence score has been computed and the matching pairs have been selected, our system makes it possible to crowdsource some of the results and aggregates them into a final matching decision.

A standard approach to aggregate the results from the crowd is *majority voting*: The 5 automatically selected candidate matchings are all proposed to 5 different workers who have to decide which matching is correct for the given instance. After the task is completed, the matching with most votes is selected as valid matching. Instead, the approach used by ZenCrowd is to aggregate the crowd results by means of the probabilistic network described in Sect. 6.

Table 2 shows the effectiveness values of the crowd on all the matching pairs in our testset. Table 3 shows the effectiveness values of the automatic approaches and their combinations with the crowd results based on both majority voting and ZenCrowd.

From Table 2, we observe that (1) the crowd performance improves by using the molecule interface, that is, displaying

Table 2 Crowd matching precision over two different HIT design interfaces (label-only and molecule) and two different aggregation methods (majority voting and ZenCrowd)

HIT	Aggregation	Organizations	People	Locations
Label-only	Maj.Vote	0.67	0.70	0.65
	ZenCrowd	0.77	0.75	0.73
Molecule	Maj.Vote	0.74	0.85	0.73
	ZenCrowd	0.81	0.87	0.81

Table 3 Matching precision for purely automatic and hybrid human/machine approaches

	Organizations	People	Locations
Inverted Index Baseline	0.78	0.98	0.89
Majority Vote	0.87	0.98	0.96
ZenCrowd	0.89	0.98	0.97

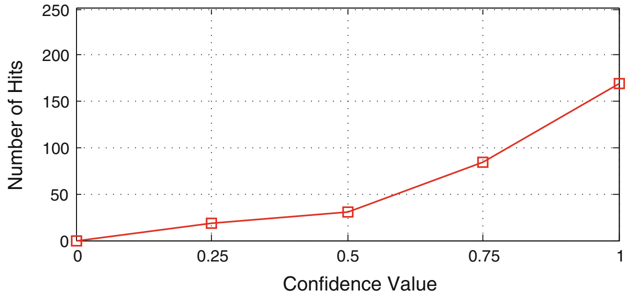


Fig. 9 Number of tasks generated for a given confidence value

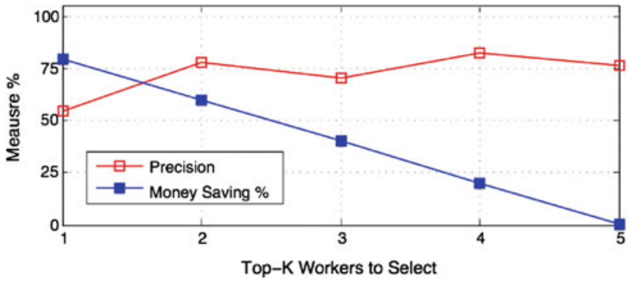


Fig. 10 ZenCrowd money saving by considering results from top-K workers only

data about the matching candidates directly from the graph database leads to higher precision consistently across different entity types as compared to the interface that only displays the instance name and lets the worker click on their link to obtain additional information; we also observe that (2) the probabilistic network used by ZenCrowd to aggregate the outcome of crowdsourcing outperforms the standard majority vote aggregation scheme in all cases.

From Table 3, we can see that ZenCrowd outperforms i) the purely automatic matching baseline based on the inverted index ranking function as well as ii) the hybrid matching approach based on automatic ranking, schema-based matching confidence, and crowdsourcing. Additionally, we observe that the most challenging type of instances to match in our experiment is organizations, while people can be matched with high precision using automatic methods only. On average over the different entity types, we could match data with a 95 % accuracy[21] (as compared to the initial 88 % average accuracy of the purely automatic baseline).

Crowdsourcing cost optimization. In addition to being interested in the effectiveness of the different matching methods, we are also interested in their cost in order to be able to select the best trade-off among the available combinations. In the following, we report on results focusing on an efficient selection of the matching pairs that the system crowdsources. After the initial blocking step based on the inverted index (that is able to filter out most of the non-relevant instances), we compute a confidence matching score for all top-ranked instances using the schema-based method. This second blocking step allows ZenCrowd to select, based on a threshold on the computed confidence score, which matching pairs to crowdsource. Setting a threshold allows to crowdsource cases with low confidence only.

Figure 9 shows how many HITs are generated by ZenCrowd by varying the threshold on the confidence score. As we can see when we set the confidence threshold to 0, then we trust completely the automatic approach and crowdsource no matching. By increasing the threshold on the matching confidence, we are required to crowdsource matchings for more than half of out testset instances. Compared to Fig. 8, we can see that the increase in the gap between precision and recall corresponds to the number of crowdsourced tasks: If recall is low, we need to crowdsource new matching tasks to obtain results about those instances the automatic approach could not match with high confidence.

Crowd performance analysis. We are also interested in understanding how the crowd performs on the instance matching task.

Figure 10 shows the trade-off between the crowdsourcing cost and the matching precision. We observe that our system is able to improve the overall matching precision by rewarding more workers (i.e., we select top-K workers based on their prior probability which is computed according to their past performance). On the other hand, it is possible to reduce the cost (as compared to the original 5 workers setup) with a limited loss in precision by considering fewer workers.

[21] This is the average accuracy over all entity types reported in Table 3.

Table 4 Correct and incorrect matchings as by crowd majority voting using two different HIT designs

Molecule	Label-only	
	Correct	Wrong
Correct	176	66
Wrong	38	20

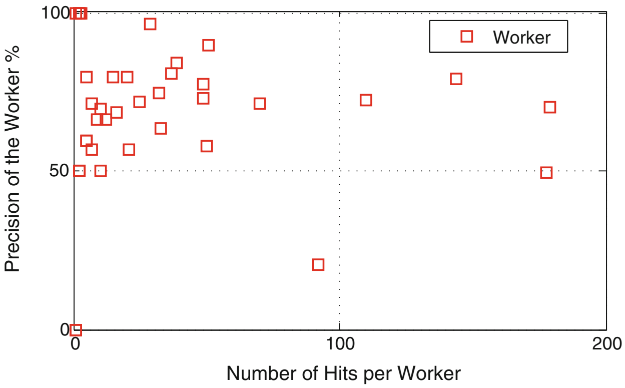


Fig. 11 Distribution of the workers' precision using the molecule design as compared to the number of tasks performed by the workers

Table 4 compares the crowd performance over the two different HIT designs. When comparing the two designs, we can observe that more errors are done with the label-only interface (i.e., 66 vs. 38) as the workers do not have much information directly on the HIT page. Interestingly, we can also see that the common errors are minimal (i.e., 20 out of 300), which motivates further analysis and possible combinations of the two designs.

Figure 11 presents the worker accuracy as compared to the number of tasks performed by the worker. As we can see, most of the workers reach precision values higher than 50 % and the workers who contributed most provide high-quality results. When compared with the worker precision over the entity linking task (see Fig. 18 top), we can see that while the power law distribution of completed HITs remains (see Fig. 17), the crowd precision on the instance matching task is clearly higher than on the entity linking task.

Efficiency. Finally, we briefly comment on the efficiency of our IM approach. In its current implementation, ZenCrowd takes on average 500 ms to select and rank candidate matchings out of the inverted index, 125 ms to obtain instance information from the graph DB, and 500 ms to generate a micro-matching task on the crowdsourcing platform. The decision process takes on average 100 ms. Without taking into account any parallelization, our system can thus offer a new matching task to the crowd roughly every second, which in our opinion is sufficient for most applications. Once on the crowdsourcing platform, the tasks have a much higher latency (several minutes to a few hours), latency, which, is however mitigated by the fact that instance matching is an embarrassingly parallel operation on crowdsourcing platforms (i.e., large collections of workers can work in parallel at any given point in time).

7.3 Discussion

Looking back at the experimental results presented so far, we first observe that crowdsourcing instance matching is useful to improve the effectiveness of an instance matching system. State-of-the-art majority voting crowdsourcing techniques can relatively improve precision up to 12 % over a purely automatic baseline (going from 0.78 to 0.87). ZenCrowd takes advantage of a probabilistic framework for making decisions and performs even better, leading to a relative performance improvement up to 14 % over our best automatic matching approach (going from 0.78 to 0.89).[22]

A more general observation is that instance matching is a challenging task, which can rapidly become impractical when errors are made at the initial blocking phases. Analyzing the population of workers on the crowdsourcing platform (see Fig. 17), we observe that the number of tasks performed by a given worker is Zipf-distributed (i.e., few workers perform many tasks, while many workers perform a few tasks only). Also, we observe that the average precision of the workers is broadly distributed between [0.5, 1] (see Fig. 11). As workers cannot be selected dynamically for a given task on the current crowdsourcing platforms (all we can do is prevent some workers from receiving any further task through blacklisting or decide not to reward workers who consistently perform bad), obtaining perfect matching results is thus in general unrealistic for non-controlled settings.

8 Experiments on entity linking

8.1 Experimental setting

Data set description. In order to evaluate ZenCrowd on the entity linking (EL) task, we created an ad hoc test collection.[23] The collection consists of 25 news articles written in English from CNN.com, NYTimes.com, washingtonpost.com, timesofindia.indiatimes.com, and swissinfo.com, which were manually selected to cover global interest news (10), US local news (5), India local news (5), and Switzerland local news (5). After the full text of the articles has been extracted from the HTML page [31], 489 entities were

[22] The improvement is statistically significant (t test $p < 0.05$).

[23] The test collection we created is available for download at: http://exascale.info/zencrowd/.

extracted from it using the Stanford Parser [30] as entity extractor. The collection of candidate URIs is composed of all entities from DBPedia,[24] Freebase,[25] Geonames,[26] and NYT,[27] summing up to approximately 40 million entities (23M from Freebase, 9M from DBPedia, 8M from Geonames, 22K from NYT). Expert editors manually selected the correct URIs for all the entities in the collection to create the ground truth for our experiments. Crowdsourcing was performed using the Amazon MTurk[28] platform where 80 distinct workers have been employed. A single task, paid $0.01, consisted of selecting the correct URIs out of the proposed five URIs for a given entity.

In the following, we present and discuss the results of a series of focused experiments, each designed to illustrate the performance of a particular feature of our EL pipeline or of related techniques. We start by describing a relatively simple base configuration for our experimental setting below.

LOD indexing, entity linking, and ranking. In order to select candidate URIs for an entity, we adopt the same IR techniques used for the IM task. We build an inverted index over 40 million entity labels in the considered LOD data sets and run queries against it using the entities extracted from the news articles in the test collection. Unless specified otherwise, the top 5 results ranked by TF-IDF are used as candidates for the crowdsourcing task.

Micro-task generation. We dynamically create a task on MTurk for each entity sent to the crowd. We generate a micro-task where the entity (possibly with some textual context) is shown to the worker who has then to select all the URIs that match the entity, with the possibility to click on the URI and visit the corresponding webpage.

If no URI matches the entity, the worker can select the "None of the above" answer. An additional field is available for the worker to leave comments.

Evaluation measures. In order to evaluate the effectiveness of our EL methods, we compare, for each entity, the selected URIs against the ground truth which provides matching/non-matching information for each candidate URI. Similarly to what we did for the IM task evaluation, we compute (P)recision, (R)ecall, and (A)ccuracy which are defined as follows: We consider as true positives (*tp*) all cases where both the ground truth and the approach select the URI, true negatives (*tn*) the cases where both the ground truth and the approach do *not* select the URI for the entity, false positives (*fp*) the cases where the approach selects a URI which is not considered correct by the ground truth, and false negatives (*fn*) the cases where the approach does non select a URI that is correct in the ground truth. Then, precision is defined as $P = tp/(tp+fp)$, recall as $R = tp/(tp+fn)$, and accuracy as $A = (tp + tn)/(tp + tn + fp + fn)$.

In the following, all the final EL approaches (automatic, agreement vote, and ZenCrowd) are optimized to return high precision values. We decided to focus on precision from the start, since from our experience, it is the most useful metric in practice (i.e., entity linking applications typically tend to favor precision to foster correct information processing capabilities and do not care whether some of the entities end up being not linked).

8.2 Experimental results

Entity extraction and linkable entities. We start by evaluating the performance of the entity extraction process. As described above, we use a state-of-the-art extractor (the Stanford Parser) for this task. According to our ground truth, 383 out of the 488 automatically extracted entities can be correctly linked to URIs in our experiments, while the remaining ones are either wrongly extracted or not available in the LOD cloud we consider. Unless stated otherwise, we average our results over all *linkable* entities, i.e., all entities for which at least one correct link can be picked out (we disregard the other entities for several experiments, since they were wrongly extracted from the text or are not at all available in the LOD data we consider and thus can be seen as a constant noise level in our experiments).

Candidate selection. We now turn to the evaluation of our candidate selection method. As described above, candidate selection consists in the present case in ranking URIs using TF-IDF given an extracted entity.[29] We focus on high recall for this phase (i.e., we aim at keeping as many potentially interesting candidates as possible) and decided to keep the top-5 URIs produced by this process. Thus, we aim at preserving as many correct URIs as possible for later linking steps (e.g., in order to provide good candidate URIs to the crowd). We report on the performance of candidate selection in Table 5.

As we can observe, results are consistent with our goal since all interesting candidates are preserved by this method (recall of 1 for the linkable entities set).

Then, we examine the potential role of the highest confidence scores in the candidate selection process. This analysis helps us decide when crowdsourcing an EL task is useful and

[24] http://dbpedia.org/.

[25] http://www.freebase.com/.

[26] http://www.geonames.org/.

[27] http://data.nytimes.com/.

[28] http://www.mturk.com.

[29] Our approach is hence similar to Blanco et al. [7], though we do not use BM25F as a ranking function.

Table 5 Performance results for the candidate selection approach

	All entities		Linkable entities	
	P	R	P	R
GL news	0.27	0.67	0.40	1.0
US news	0.17	0.46	0.36	1.0
IN news	0.22	0.62	0.36	1.0
SW news	0.21	0.63	0.34	1.0
All news	0.24	0.63	0.37	1.0

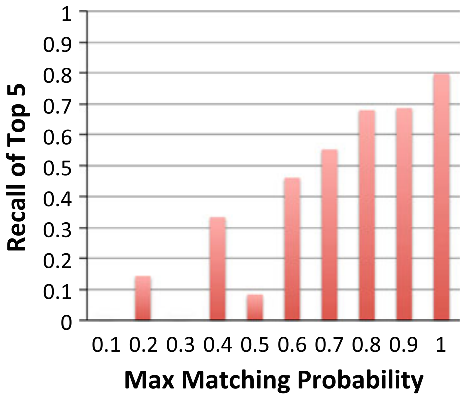


Fig. 12 Average recall of candidate selection when discriminating on maximum relevance probability in the candidate URI set

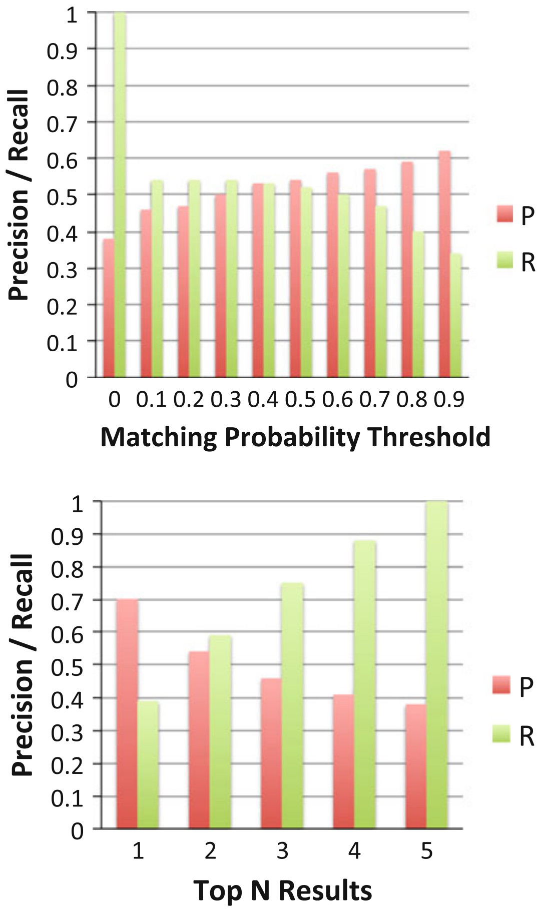


Fig. 13 Performance results (precision, recall) for the automatic approach

when it is not. In Fig. 12, we report on the average recall of the top-5 candidates when classifying results based on the maximum confidence score obtained (top-1 score). The results are averaged over *all* extracted entities.[30]

As expected, we observe that high confidence values for the candidates selection lead to high recall and, therefore, to candidate sets which contain many of the correct URIs. For this reason, it is useful to crowdsource EL tasks only for those cases exhibiting relatively high confidence values (e.g., >0.5). When the highest confidence value in the candidate set is low, it is then more likely that no URI will match the entity (because the entity has no URI in the LOD cloud we consider, or because the entity extractor extracted the entity wrongly).

On the other hand, crowdsourcing might be unnecessary for cases where the precision of the automatic candidate selection phase is already quite high. The automatic selection techniques can be adapted to identify the correct URIs in a completely automatic fashion. In the following, we automatically select top-1 candidates only (i.e., the link with the highest confidence), in order to focus on high precision results as required by many practical applications. A different approach focusing on recall might select all candidates with a confidence higher than a certain threshold. Fig. 13 reports on the performance of our fully automatic entity linking approaches. We observe that when the top-1 URI is selected, the automatic approach reaches a precision value of 0.70 at the cost of low recall (i.e., fewer links are picked). As latter results will show, crowdsourcing techniques can improve both precision and recall over this automatic entity linking approaches in all cases.

Entity linking using crowdsourcing with agreement vote. We now report on the performance of a state-of-the-art crowdsourcing approach based on agreement voting: The 5 automatically selected candidate URIs are all proposed to 5 different workers who have to decide which URI(s) is (are) correct for the given entity. After the task is completed, the URIs with at least 2 votes are selected as valid links (we tried various thresholds and manually picked 2 in the end since it leads to the highest precision scores while keeping good recall values for our experiments). We report on the performance of this crowdsourcing technique in Table 6. The values

[30] Confidence scores have all been normalized to [0, 1] by manually defining a transformation function.

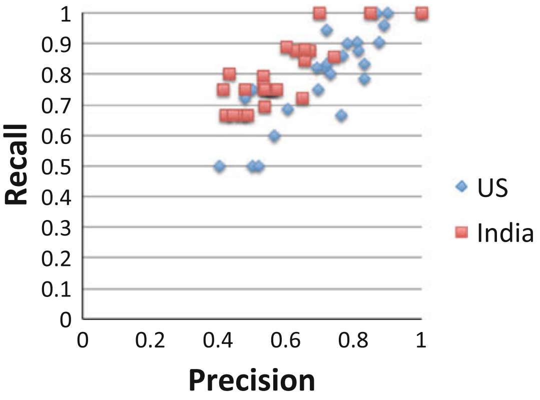


Fig. 14 Per document task effectiveness

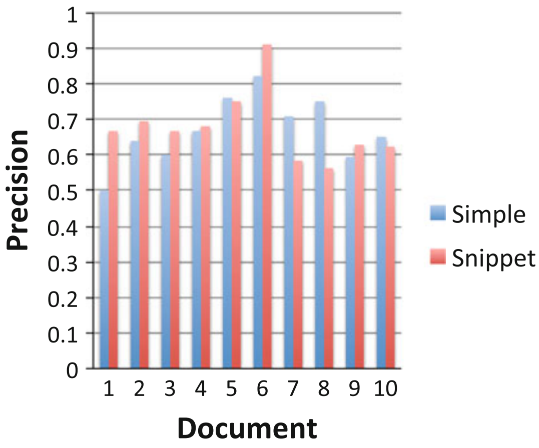


Fig. 15 Crowdsourcing results with two different textual contexts

are averaged over all linkable entities for different document types and worker communities.

The first question we examine is whether there is a difference in reliability between the various populations of workers. In Fig. 14, we show the performance for tasks performed by workers located in USA and India (each point corresponds to the average precision and recall over all entities in one document). On average, we observe that tasks performed by workers located in the USA lead to higher precision values. As we can see in Table 6, Indian workers obtain higher precision and recall on local Indian news as compared to US workers. The biggest difference in terms of accuracy between the two communities can be observed on the global interest news.

A second question we examine is how the textual context given for an entity influences the worker performance. In Fig. 15, we compare the tasks for which only the entity label is given (simple) to those for which a context consisting of all the sentences containing the entity are shown to the worker (snippets). Surprisingly, we could not observe a significant difference in effectiveness caused by the different textual contexts given to the workers. Thus, we focus on only one type of context for the remaining experiments (we always give the snippet context).

Entity linking with ZenCrowd. We now focus on the performance of the probabilistic inference network as proposed in this paper. We consider the method described in Sect. 6, with an initial training phase consisting of 5 entities and a second, continuous training phase, consisting of 5 % of the other entities being offered to the workers (i.e., the workers are given a task whose solution is known by the system every 20 tasks on average).

In order to reduce the number of tasks having little influence in the final results, a simple technique of blacklisting of bad workers is used. A bad worker (who can be considered as a *spammer*) is a worker who randomly and rapidly clicks on the links, hence generating noise in our system. In our experiments, we consider that 3 consecutive bad answers in the training phase is enough to identify the worker as a spammer and to blacklist him/her. We report the average results of ZenCrowd when exploiting the training phase, constraints, and blacklisting in Table 7. As we can observe, precision and accuracy values are higher in all cases when compared to the agreement vote approach (see Table 6).

Finally, we compare ZenCrowd to the state-of-the-art crowdsourcing approach (using the optimal agreement vote) and our best automatic approach on a per-task basis in Fig. 16. The comparison is given for each document in the test collection. We observe that in most cases, the Human Intelligence contribution improves the precision of the automatic approach. We also observe that ZenCrowd dominates the overall performance (it is the best performing approach in more than 3/4 of the cases).

Efficiency. Finally, we briefly comment on the efficiency of our approach. In its current implementation, ZenCrowd takes on average 200 ms to extract an entity from text, 500 ms to select and rank candidate URIs, and 500 ms to generate a micro-linking task. The decision process takes on average 100 ms. The same observations about parallelization on crowdsourcing platforms already done for the IM task hold for the EL task as well.

8.3 Discussion

Looking at the experimental results about the EL task presented above, we observe that the crowdsourcing step improves the overall EL effectiveness of the system.

Standard crowdsourcing techniques (i.e., using agreement vote aggregation) yield a relative improvement of 6 % in precision (from 0.70 to 0.74). ZenCrowd, by leveraging the probabilistic framework for making decisions, performs better, leading to a relative performance improvement ranging

Table 6 Performance results for crowdsourcing with agreement vote over linkable entities

	US workers			Indian workers		
	P	R	A	P	R	A
GL news	0.79	0.85	0.77	0.60	0.80	0.60
US news	0.52	0.61	0.54	0.50	0.74	0.47
IN news	0.62	0.76	0.65	0.64	0.86	0.63
SW news	0.69	0.82	0.69	0.50	0.69	0.56
All news	0.74	0.82	0.73	0.57	0.78	0.59

Table 7 Performance results for crowdsourcing with ZenCrowd over linkable entities

	US workers			Indian workers		
	P	R	A	P	R	A
GL news	0.84	0.87	0.90	0.67	0.64	0.78
US news	0.64	0.68	0.78	0.55	0.63	0.71
IN news	0.84	0.82	0.89	0.75	0.77	0.80
SW news	0.72	0.80	0.85	0.61	0.62	0.73
All news	0.80	0.81	0.88	0.64	0.62	0.76

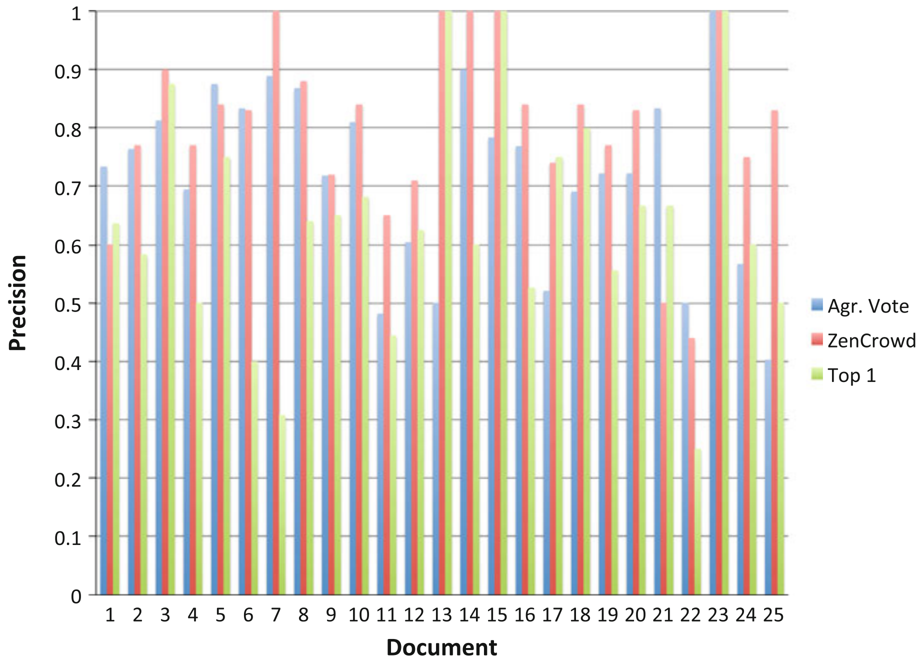


Fig. 16 Comparison of three linking techniques

between 4 and 35 % over the agreement vote approach and on average of 14 % over our best automatic linking approach (from 0.70 to 0.80). In both cases, the improvement is statistically significant (t-test $p < 0.05$).

Analyzing worker activities on the crowdsourcing platform (see Fig. 17), we observe that the number of tasks performed by a given worker is Zipf-distributed (i.e., few workers perform many tasks, while many workers perform a few tasks only).

Augmenting the numbers of workers performing a given task is not always beneficial: Figure 18, bottom, shows how the average precision of ZenCrowd varies when (virtually) employing the available top-k workers for a given task. As can be seen from the graph, the quality of the results gets worse after a certain value of k, as more and more mediocre workers are picked out. As a general rule, we observe that limiting the number of workers to 4 or 5 good workers for a given task gives the best results.

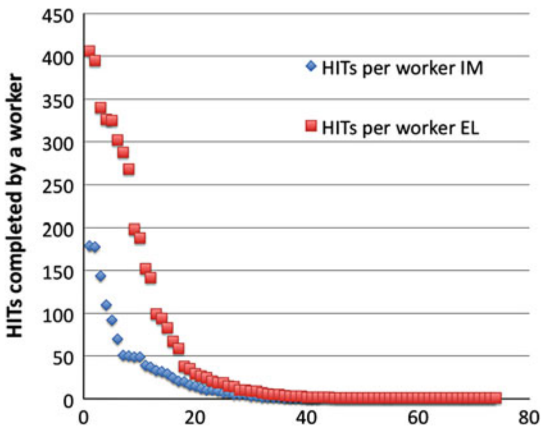


Fig. 17 Number of HITs completed by each worker for both IM and EL ordered by most productive workers first

The intuition behind using the probabilistic network is that a worker who proves that he is good, i.e., has a high prior probability, should be trusted for future jobs. Furthermore, his/her answer should always prevail and help identifying other good workers. Also, the probabilistic network takes advantage of constraints to help the decision process.

While the data sets used for the IM and EL evaluations are different, we can make some observation on the average effectiveness reached for each task. On average, the effectiveness of the workers on the IM task is higher than that on the EL task. However, we observe that ZenCrowd is able to exploit the work performed by the most effective workers (e.g., top US worker in Fig. 18 top or the highly productive workers in Fig. 11).

9 Conclusions

As the LOD movement gains momentum, matching instances across data sets and linking traditional Web content to the LOD cloud is getting increasingly important in order to foster automated information processing capabilities. Current tools rely either on fully automated techniques or on the sole work of human experts. In this paper, we have presented ZenCrowd, a data integration system based on a probabilistic framework leveraging both automatic techniques and punctual human intelligence feedback captured on a crowdsourcing platform. ZenCrowd adopts a novel three-stage blocking process that can deal with very large data sets while at the same time minimizing the cost of crowdsourcing by carefully selecting the right candidate matches to crowdsource.

As our approach incorporates a human intelligence component, it typically cannot perform instance matching and entity linking tasks in real time. However, we believe that it can still be used in most practical settings, thanks to the embarrassingly parallel nature of data integration in crowdsourcing environments.

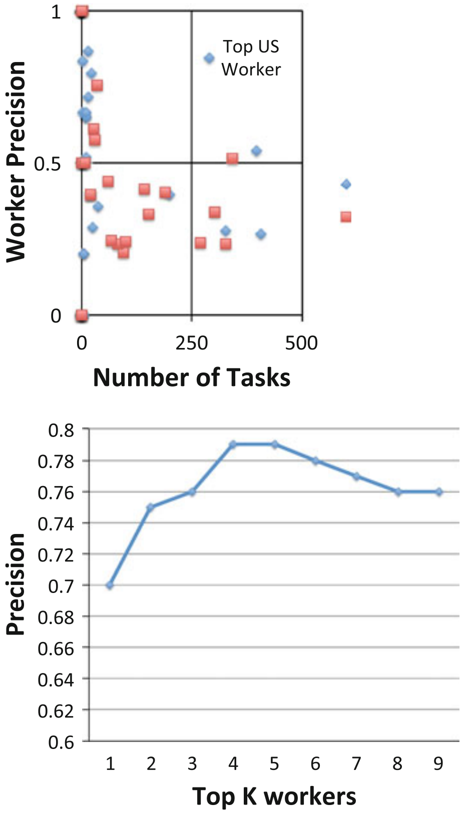


Fig. 18 Distribution of the workers' precision for the entity linking task as compared to the number of tasks performed by the worker (*top*) and task precision with top *k* workers (*bottom*)

In conclusion, ZenCrowd provides a reliable approach to entity linking and instance matching, which exploits the trade-off between *large-scale* automatic instance matching and *high-quality* human annotation, and which according to our results improves the precision of the instance matching results up to 14 % over our best automatic matching approach for the instance matching task. For the entity linking task, ZenCrowd improves the precision of the results by 4–35 % over a state of the art and manually optimized crowdsourcing approach, and on average by 14 % over our best automatic approach.

Possible future directions include the analysis of how similar documents linking to different entities can provide further indication on how instances could match. Moreover, considering documents written in languages other than English could be addressed by exploiting the multilingual property of many LOD data sets. Another potential extension is the comparison of different HIT designs for the instance matching and entity linking tasks using, for instance, images instead of textual entity descriptions.

References

1. Alonso, O., Baeza-Yates, R.A.: Design and implementation of relevance assessments using crowdsourcing. In: ECIR, pp. 153–164 (2011).
2. Bailey, P., de Vries, A.P., Craswell, N., Soboroff, I.: Overview of the TREC 2007 enterprise track. In: TREC (2007)
3. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2010 entity track. In: TREC (2010)
4. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: IJCAI, pp. 2670–2676 (2007)
5. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '03, pp. 39–48. ACM, New York (2003). doi:10.1145/956750.956759
6. Blanco, R., Halpin, H., Herzig, D., Mika, P., Pound, J., Thompson, H.S., Tran, D.T.: Repeatable and reliable search system evaluation using crowdsourcing. In: SIGIR, pp. 923–932 (2011)
7. Blanco, R., Mika, P., Vigna, S.: Effective and efficient entity search in RDF data. In: International Semantic Web Conference (ISWC), pp. 83–97 (2011)
8. Bouquet, P., Stoermer, H., Niederee, C., Mana, A.: Entity name system: the backbone of an open and scalable web of data. In: Proceedings of the IEEE International Conference on Semantic Computing (ICSC), pp. 554–561 (2008)
9. Bunescu, R., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: Proceedings of EACL, vol. 6 (2006)
10. Bunescu, R.C., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: EACL (2006)
11. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. **24**(9), 1537–1555 (2012). doi:10.1109/TKDE.2011.127
12. Ciaramita, M., Altun, Y.: Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, pp. 594–602. ACL, Stroudsburg (2006). http://dl.acm.org/citation.cfm?id=1610075.1610158
13. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: Proceedings of EMNLP-CoNLL, vol. 2007, pp. 708–716 (2007)
14. Cudré-Mauroux, P., Aberer, K., Feher, A.: Probabilistic message passing in peer data management systems. In: International Conference on Data Engineering (ICDE) (2006)
15. Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., De Meer, H.: idMesh: graph-based disambiguation of linked data. In: WWW '09, pp. 591–600. ACM, New York (2009). doi:10.1145/1526709.1526789
16. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Anniversary Meeting of the ACL (2002)
17. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: Proceedings of the 21st International Conference on World Wide Web, WWW '12, pp. 469–478. ACM, New York (2012). doi:10.1145/2187836.2187900
18. Demartini, G., Iofciu, T., de Vries, A.P.: Overview of the INEX 2009 entity ranking track. In: INEX, pp. 254–264 (2009)
19. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. 39 (1977)
20. Difallah, D.E., Demartini, G., Cudré-Mauroux, P.: Pick-A-Crowd: Tell me what you like, and I'll tell you what to do. In: WWW'13. ACM, New York (2013)
21. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD, pp. 85–96. ACM, New York (2005)
22. Feng, A., Franklin, M.J., Kossmann, D., Kraska, T., Madden, S., Ramesh, S., Wang, A., Xin, R.: CrowdDB: Query Processing with the VLDB Crowd. PVLDB **4**(11), 1387–1390 (2011)
23. Finin, T., Murnane, W., Karandikar, A., Keller, N., Martineau, J., Dredze, M.: Annotating named entities in Twitter data with crowdsourcing. In: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, CSLDAMT '10, pp. 80–88 (2010)
24. Getoor, L., Machanavajjhala, A.: Entity resolution: tutorial. In: VLDB (2012)
25. Haas, K., Mika, P., Tarjan, P., Blanco, R.: Enhanced results for web search. In: SIGIR, pp. 725–734 (2011)
26. Han, X., Zhao, J.: Named entity disambiguation by leveraging wikipedia semantic knowledge. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, pp. 215–224. ACM, New York (2009). doi:10.1145/1645953.1645983
27. Jaro, M.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa. Florida. J. Am. Stat. Assoc. **84**(406), 414–420 (1989)
28. Kazai, G.: In search of quality in crowdsourcing for search engine evaluation. In: ECIR, pp. 165–176 (2011)
29. Kazai, G., Kamps, J., Koolen, M., Milic-Frayling, N.: Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In: SIGIR, pp. 205–214 (2011)
30. Klein, D., Manning, C.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, vol. 1, pp. 423–430 (2003)
31. Kohlschütter, C., Fankhauser, P., Nejdl, W.: Boilerplate detection using shallow text features. In: WSDM, pp. 441–450 (2010)
32. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inform. Theory 47(2) (2001)
33. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol. 10, pp. 707–710 (1966)
34. Lim, E.P., Srivastava, J., Prabhakar, S., Richardson, J.: Entity identification in database integration. Inform. Sci. **89**(12), 1–38 (1996). doi:10.1016/0020-0255(95)00185-9
35. Liu, X., Lu, M., Ooi, B.C., Shen, Y., Wu, S., Zhang, M.: Cdas: a crowdsourcing data analytics system. Proc. VLDB Endow. 5(10), 1040–1051 (2012). http://dl.acm.org/citation.cfm?id=2336664.2336676
36. Marcus, A., Wu, E., Karger, D.R., Madden, S., Miller, R.C.: Human-powered sorts and joins. PVLDB **5**(1), 13–24 (2011)
37. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems (I-Semantics) (2011)
38. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pp. 233–242. ACM, New York (2007). doi:10.1145/1321440.1321475
39. Murphy, K.M., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: an empirical study. In: Uncertainty in Artificial Intelligence (UAI) (1999)
40. On, B., Koudas, N., Lee, D., Srivastava, D.: Group linkage. In: Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE), pp. 496–505 (2007)

41. Papadakis, G., Ioannou, E., Niederée, C., Palpanas, T., Nejdl, W.: Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, pp. 53–62. ACM, New York (2012). doi:10.1145/2124295.2124305
42. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: WWW, pp. 771–780 (2010)
43. Selke, J., Lofi, C., Balke, W.T.: Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. Proc. VLDB Endow. 5(6), 538–549 (2012). http://dl.acm.org/citation.cfm?id=2168651.2168655
44. Shen, W., Wang, J., Luo, P., Wang, M.: Liege: link entities in web lists with knowledge base. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12, pp. 1424–1432. ACM, New York (2012). doi:10.1145/2339530.2339753
45. Tonon, A., Demartini, G., Cudré-Mauroux, P.: Combining inverted indices and structured search for ad-hoc object retrieval. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, pp. 125–134. ACM, New York (2012). doi:10.1145/2348283.2348304
46. von Ahn, L., Dabbish, L.: Designing games with a purpose. Commun. ACM **51**(8), 58–67 (2008). doi:10.1145/1378704.1378719
47. von Ahn, L., Liu, R., Blum, M.: Peekaboom: a game for locating objects in images. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06, pp. 55–64. ACM, New York (2006). doi:10.1145/1124772.1124782
48. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: CrowdER: crowdsourcing entity resolution. PVLDB **5**(11), 1483–1494 (2012)
49. Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, SIGMOD '09, pp. 219–232. ACM, New York (2009). doi:10.1145/1559845.1559870
50. Winkler, W.: The state of record linkage and current research problems. US Census Bureau. In: Statistical Research Division (1999)
51. Wylot, M., Pont, J., Wisniewski, M., Cudré-Mauroux, P.: dipLODocus[RDF]—short and long-tail rdf analytics for massive webs of data. In: International Semantic Web Conference (ISWC), pp. 778–793 (2011)