ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux
eXascale Infolab
U. of Fribourg—Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

We tackle the problem of entity linking for large collections of online pages; Our system, ZenCrowd, identifies entities from natural language text using state of the art techniques and automatically connects them to the Linked Open Data cloud. We show how one can take advantage of human intelligence to improve the quality of the links by dynamically generating micro-tasks on an online crowdsourcing platform. We develop a probabilistic framework to make sensible decisions about candidate links and to identify unreliable human workers. We evaluate ZenCrowd in a real deployment and show how a combination of both probabilistic reasoning and crowdsourcing techniques can significantly improve the quality of the links, while limiting the amount of work performed by the crowd.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; D.2.12.a [**Software Engineering**]: Interoperability—*Data Mapping*

Keywords

Entity Linking, Linked Data, Crowdsourcing, Probabilistic Reasoning

1. INTRODUCTION

The Linked Open Data (LOD) community is currently bringing structured data to the Web by publishing data sets using RDF and by interlinking related concepts coming from different data sets. 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, linking textual content to LOD concepts opens the door to automated text enrichment (e.g., by providing additional information coming from the LOD cloud for the entities appearing in the text), as well as to streamlined information retrieval and integration (e.g., by using links to retrieve all text articles related to a given concept from the LOD cloud).

Automatizing the process of extracting entities from natural language text and linking those entities to the correct structured concept(s) in the LOD cloud is currently drawing a lot of attention (see the Related Work section below). It represents however a daunting task, as entity matching is known to be extremely challenging even in relatively simple contexts, since parsing and disambiguating natural language text is still extremely difficult for machines nowadays.

The current matching techniques used to relate an entity extracted from text to corresponding entities from the LOD cloud can be broadly classified into two groups:

Algorithmic Matching: Given the scale of the problem (that could potentially span the entire HTML Web), many efforts are currently focusing on designing and deploying scalable algorithms to perform the matching automatically on very large corpuses.

Manual Matching: While algorithmic matching techniques are constantly improving, they are still at this stage not as reliable as humans. Hence, many organizations are still today appointing individuals to manually link textual elements to concepts. For instance, the New York Times employs a whole team whose sole responsibility is to manually create links from news articles to NYT identifiers[1].

This paper represents a step towards bridging the gap between those two classes of techniques by parsimoniously using human workers to guide the automated linking process. We introduce a new system, called *ZenCrowd*, which gracefully combines algorithmic and manual matching. ZenCrowd takes advantage of algorithmic matching techniques to routinely link entities, but attempts to improve the automatic results by involving human workers. Our solution systematizes and automatizes manual matching techniques by dynamically creating *micro* matching tasks and by publishing them on a popular crowdsourcing platform. To operate the system efficiently, we develop a probabilistic framework to decide how to incorporate manual matching, and to more effectively integrate inconsistent results obtained by arbitrary sets of human workers on the crowdsourcing platform.

ZenCrowd does *not* focus on the algorithmic entity extraction or entity matching problems *per se* (we use state of the art techniques for both automated entity extraction and entity matching, but do not directly innovate on that front). However, we believe that we make a number of key contributions at the interface of algorithmic and manual matching, and discuss in detail how to most effectively and efficiently combine the two approaches using both theoretical models and experimental results. The contributions of this paper include:

- a new system architecture supporting both algorithmic and manual matching approaches in concert

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

[1] see `http://data.nytimes.com/`

- new techniques to automate manual matching tasks taking advantage of scoping mechanisms and modern crowdsourcing platforms
- new techniques to evaluate the combined performance of algorithmic and manual matching on sets of entities
- a new probabilistic reasoning 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 algorithmic matching, uniqueness constraints, and identity links from the LOD cloud
- an empirical evaluation of our system in a real deployment over different countries showing that ZenCrowd combines the best of both worlds, in the sense that our combined approach turns out to be more effective than both algorithmic and manual matching techniques for online entity linking.

The rest of this paper is structured as follows: We review the state of the art in entity linking, entity matching and crowdsourcing systems in Section 2. Section 3 gives an overview of the architecture of our system, including its algorithmic matching interface, its probabilistic reasoning engine, and its templating and crowdsourcing components. We describe our formal model to combine both algorithmic and crowdsourcing results using probabilistic reasoning in Section 4. We introduce our evaluation methodology and discuss results from a real deployment of our system in Section 5, before concluding in Section 6.

2. RELATED WORK

Entity Linking. Entities have recently become first-class citizens on the Web. A large amount of online search queries are about entities [29], and search engines exploit entities and structured data to build their result pages [17]. In the field of Information Retrieval (IR) a lot of attention has been given to entities: At TREC[2], 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 evaluated [12] Entity Ranking in Wikipedia at INEX[3] recently.

The problem of assigning URIs to entities (i.e., entity linking), which is the focus of our paper, 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[4], which suggested the idea of an Entity Name System (ENS) to assign identifiers to entities on the Web [7]. 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 [8, 4]. State of the art techniques are implemented in tools like Gate [11], the Stanford parser [21] (which we use in our experiments), and Extractiv[5]. 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"). Classical matching approaches are based on string similarities ("Barack Obama" vs. "B. Obama") such as the edit distance [24], the Jaro similarity [18], or the Jaro-Winkler similarity [30]. More advanced techniques, as for instance Group Linkage [28], compare groups of records to find matches. A third class of approaches uses semantic information. Reference Reconciliation [14], for example, builds a dependency graph and exploits relations to propagate information among entities. In [10], we build disambiguation graphs based on the transitive closures of equivalence links in networks containing uncertain information. Our present work focuses on a very different topic and aims at correctly linking isolated entities to external entities using an effective combination of algorithmic and manual matching techniques. 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[6], DBPedia Spotlight [26]). Relevant work aims for instance at enriching documents by automatically creating links to Wikipedia pages [27], which can be seen as entity identifiers. While previous work selects URIs from a specific corpus (e.g., DBPedia, Wikipedia), our goal is to assign entity identifiers from the LOD cloud[7] instead.

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.

Ad-Hoc Object Retrieval. Another task related to the one we are addressing in this paper is Ad-hoc Object Retrieval (AOR) [29], 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 workshop in 2010 and 2011 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 [6]) exploit IR indexing and ranking techniques over the RDF dataset used at the Billion Triple Challenge 2009. Similarly to such tasks, our dataset is composed of a large set of triples coming from LOD datasets, while our queries consist of entities extracted from news articles and 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.

Crowdsourcing. 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, 19, 20]. 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 [5]. The database community is currently evaluating how crowdsourcing methods can be used to build RDMS systems able to answer complex queries where subjective comparison are needed (e.g., "10 papers with the most novel ideas") [15]. Crowdsourcing can also be used for basic computational operations such as sort and join [25].

[2] `http://trec.nist.gov`

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

[4] `http://www.okkam.org`

[5] `http://extractiv.com/`

[6] `http://www.opencalais.com/`

[7] `http://linkeddata.org/`

In the context of entity identification, crowdsourcing has been used by Finn *et al* [16] 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 amounts of entities on the Web. The two approaches might be combined to obtain high quality for both extraction and linking.

3. ARCHITECTURE

ZenCrowd is a hybrid platform that takes advantage of both algorithmic and manual matching techniques simultaneously. Figure 1 presents a simplified architecture of our system. We start by giving an overview of our system below in Section 3.1, and then describe in more detail some of its components in Sections 3.2 to 3.5.

3.1 System Overview

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 on the page. Once detected, the entities are fed into *Algorithmic Matchers* that attempt to automatically link the textual entities to semantically similar entities from the LOD cloud. As querying the Web of data dynamically to match each entity would incur a very high latency, we build a local cache (called *LOD Index* in Figure 1) to locally retrieve and index relevant information from the LOD cloud. Algorithmic matchers return lists of top-k *links* to LOD entities, along with a confidence value for each potentially relevant link.

The results of the algorithmic matchers are stored in a *Probabilistic Network*, and are then combined and analyzed using probabilistic inference techniques. ZenCrowd treats the results of the algorithmic matchers 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 matchers have a low confidence value), the results get discarded. Finally, if the results are deemed promising but uncertain (for example because several algorithmic matchers 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-matching task using a templating engine. Once created, the micro-matching 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 matching results are available for a given HTML page, an enriched HTML page—containing both the original HTML code as well as RDFa annotations linking the textual entities to their counterpart from the LOD cloud—is finally generated.

3.2 Extractors & Algorithmic Matchers

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 matchers, whose role is to find semantically similar entities from the LOD cloud. ZenCrowd implements a number of state of the art matching techniques (see Section 5 for an example) 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 below Section 4).

3.3 LOD Index

The LOD index is a declarative information retrieval engine used to speed up the matching 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 that they impose. Instead of querying the LOD cloud dynamically for each new entity, 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. The information thus extracted is cached locally in two ways: in our efficient analytical query engine [31]—offering a SPARQL interface—and in an inverted index to provide efficient support for unstructured queries.

3.4 Probabilistic Graph & Decision Engine

Instead of using heuristics or arbitrary rules, ZenCrowd systematizes the use of probabilistic networks to make sensible decisions about the entities. All evidences gathered both from the algorithmic matchers 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 Section 4.

3.5 Micro-Task Manager

The micro-task manager, finally, is responsible for dynamically creating human computation tasks that are then published on a crowdsourcing platform. Whenever an entity 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 textual entity ii) some contextual information generated using a template and the original HTML document from which the entity was extracted and iii) the current top-k matches for the entity from the Probabilistic Network. We experimented with various templates to extract contextual information from the HTML pages, including *named entity* providing just the text corresponding to the extracted entity, and *text snippet* extracting the surrounding text around each occurrence of the entity in the HTML page (we report findings for those two templates in Section 5). 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 links

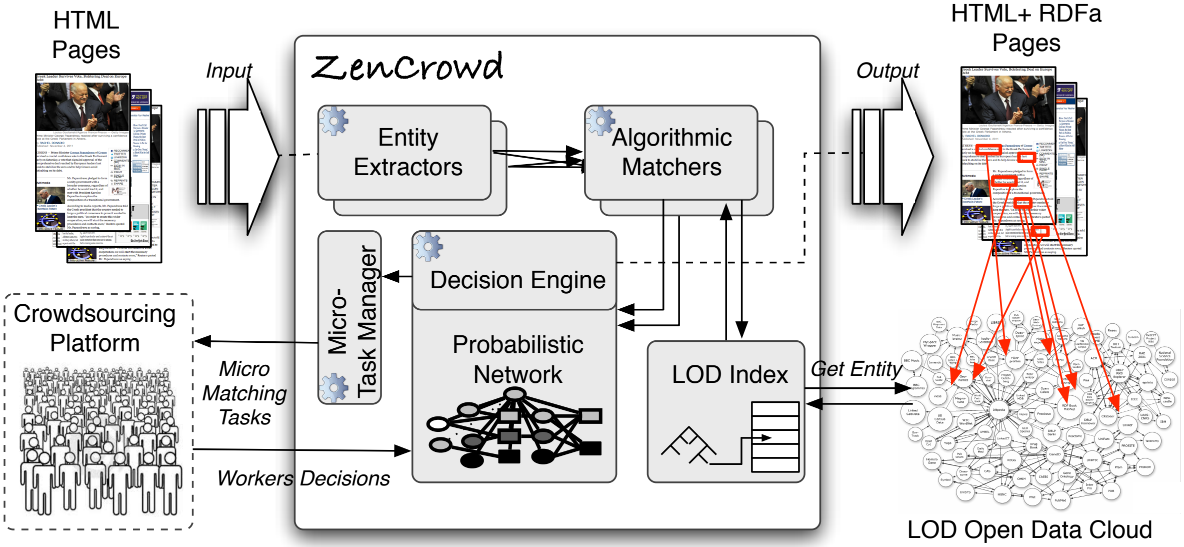

Figure 1: The architecture of ZenCrowd: Our system takes as input a collection of HTML pages and enrich them by extracting textual entities appearing in the pages and linking them to the Linked Open Data cloud. ZenCrowd uses both algorithmic matchers and human workers to generate high quality results.

(if any) for the entity, given its name, the contextual information from the original HTML text, and the various candidate matches linking to their online description in the LOD cloud. Once performed, the results of the micro-matching tasks are sent back to the Micro-Task Manager, which in turns inserts them in the Probabilistic Network.

4. PROBABILISTIC MODELS

ZenCrowd exploits probabilistic models to make sensible decisions about candidate matches. 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.

4.1 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 [23]. 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 Figure 2, where variables (circles) are linked to their respective factors (black squares).

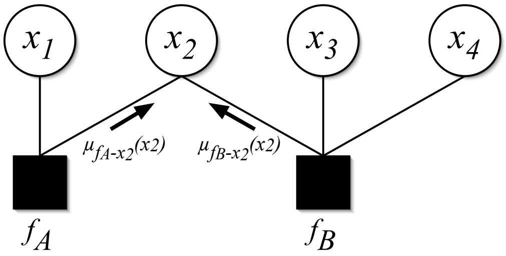

Figure 2: A simple factor-graph of four variables and two factors.

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

$$g_2(x_2) = \sum_{x_1}\sum_{x_3}\sum_{x_4} g(x_1, x_2, x_3, x_4) = \sum_{\sim\{x_2\}} g(x_1, x_2, x_3, x_4)$$

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:

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

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 Figure 2). The *sum-product* algorithm [23] exploits this observation to compute all marginal functions of a factor-graph in a concurrent and efficient manner.

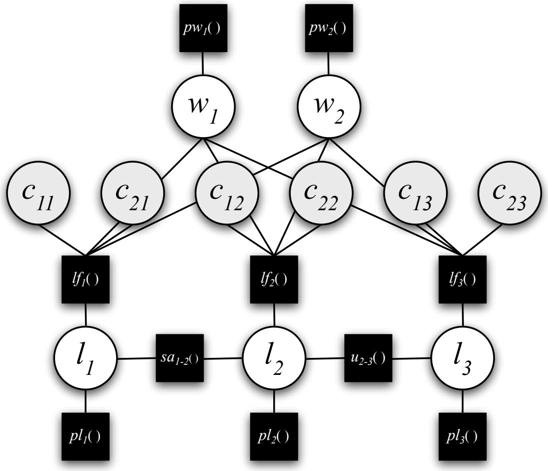

Figure 3: An entity factor-graph connecting two workers (w_i), six clicks (c_ij), and three candidate links (l_j).

4.2 Entity Graphs

We start by describing the probabilistic graphs used to combine all evidences gathered for a given entity. Consider an entity e extracted from an HTML page. Our probabilistic graph stores all candidate matches for the entity coming from the LOD cloud. The candidate matches are stored as a list of potential links l_j linking to the LOD cloud. Each link has a prior probability distribution pl_j computed from the algorithmic matchers. Each link 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 link corresponds (or not) to the entity from his/her perspective.

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

A simple example of such an entity graph is given in Figure 3. Clicks, workers, and links are further connected through three factors described below.

4.2.1 Link Factors

Link factors $lf_j()$ connect each link 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: i) clicks from reliable workers should weight more than clicks from unreliable workers (actually, clicks from consistently unreliable workers deciding randomly if a given link is relevant or not should have no weight at all in our decision process) and ii) when reliable workers do not agree, the likelihood of the link being correct should be proportional to the fraction of good workers indicating the link 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:

$$lf(w_1, \ldots, w_m, c_1, \ldots, c_n, l) =$$
$$\begin{cases} 0.5 & \text{if } \forall w_i \in \{w_1, \ldots, w_m\}\ w_i = Bad \\ \frac{\sum_i \mathbb{1}_{(w_i=Good\ \wedge\ c_i=l)}}{\sum_i \mathbb{1}_{(w_i=Good)}} & \end{cases}$$

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

4.2.2 SameAs Constraints

SameAs constraints exploit the fact that the resources identified by the links to the LOD cloud can themselves be interlinked (e.g., *http://dbpedia.org/resource/Fribourg* is connected through an *owl:sameAs* link to *fbase:Fribourg* in the LOD cloud). Considering that the *SameAs* links are correct, we define a constraint on the variables connected by such links; the factor $sa()$ connecting those variables puts a constraint forbidding assignments where the variables would not be set to the same values:

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

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 [10]) can be put into place.

4.2.3 Unicity Constraints

Many LOD datasets are curated manually, or are generated from manually-curated databases such as Wikipedia. In those datasets, a given entity is represented by exactly one URI (i.e., the datasets do not contain duplicate entities). When several links from such a dataset appear in an entity graph, we can thus rule out all configurations where more than one of those links are considered as $Correct$. The corresponding factor $u()$ is declared as being equal to 1 and is defined as follows:

$$u(l_1, \ldots, l_n) = \begin{cases} 0 & \text{if } \exists (l_i, l_j) \in \{l_1, \ldots, l_n\} | l_i = l_j = Correct \\ 1 & \text{otherwise} \end{cases}$$

4.3 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 links 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.

4.4 Updating 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 links, 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 link 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 [9, 13] 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 $P(w = reliable|l_i = Correct/Incorrect)$ as soon as a decision is reached on a link. Treat these evidences as new observations on the reliability of the workers, and update their prior beliefs iteratively as follows:

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

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 links. This technique can also be used to identify and *blacklist* unreliable workers dynamically (see Section 5.2 for an illustration).

4.5 Selective Model Instantiation

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

5. EXPERIMENTS

5.1 Experimental Setting

Dataset Description.

In order to evaluate ZenCrowd, we created an ad-hoc test collection[8]. 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 [22], 489 entities were extracted from it using the Stanford Parser [21] as entity extractor. The collection of candidate URIs is composed of all entities from DBPedia[9], Freebase[10], Geonames[11], and NYT[12], 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[13] 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.

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 system or of related techniques. Though many other experiments could have been performed, we believe that the set of experiments presented below gives an particularly accurate account of the performance of ZenCrowd. We start by describing a relatively simple base-configuration for our experimental setting below.

Candidate Selection: LOD Indexing, Entity Matching and Ranking.

In order to select candidate URIs for an entity, 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 entity labels in the considered LOD datasets, 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 methods we compare, for each entity, the selected URIs against the ground truth which provides matching/non-matching information for each candidate URI. Specifically, we compute (P)recision, (R)ecall, and (A)ccuracy which are defined as follows: We consider as true positives (*tp*) all

[8] The test collection we created is available for download at: `http://diuf.unifr.ch/xi/zencrowd/`.

[9] `http://dbpedia.org/`

[10] `http://www.freebase.com/`

[11] `http://www.geonames.org/`

[12] `http://data.nytimes.com/`

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

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 matching 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 if some of the entities end up being not linked).

5.2 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 are 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[14]. 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 matching steps (e.g., in order to provide good candidate URIs to the crowd). We report on the performance of candidate selection in Table 1.

Table 1: 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

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 entity matching task is useful and when it is not. In Figure 4, 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[15].

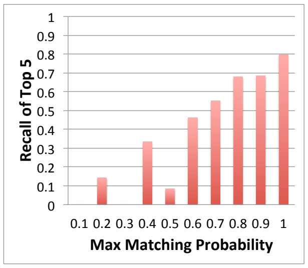

Figure 4: Average Recall of candidate selection when discriminating on max relevance probability in the candidate URI set.

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 entity matching 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. Figure 5 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 later results will show, crowdsourcing techniques can improve both precision and recall results 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,

[14] Our approach is hence similar to [6], though we do not use BM25F as a ranking function.

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

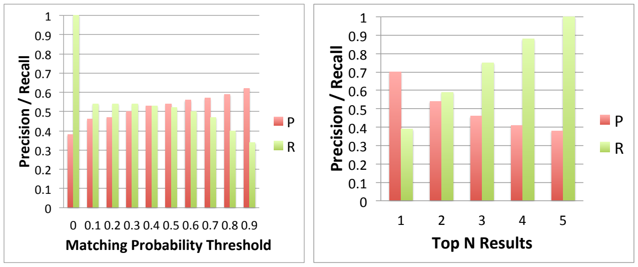

Figure 5: Performance results (Precision, Recall) for the automatic approach.

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

Table 2: 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

The first question we examine is whether there is a difference in reliability between the various populations of workers. In Figure 6 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 2, 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.

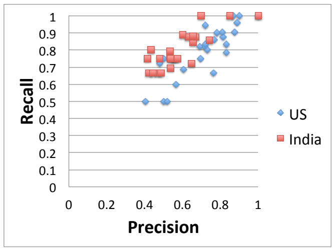

Figure 6: Per document task effectiveness.

A second question we examine is how the textual context given for an entity influences the worker performance. In Figure 7, we compare the tasks for which only the entity label is given 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).

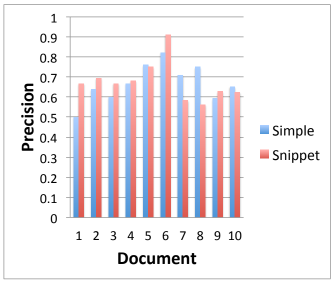

Figure 7: Crowdsourcing results with two different textual contexts

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 Section 4, 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 3. As we can observe, precision and accuracy values are higher in all cases when compared to the agreement vote approach.

Table 3: 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

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 Figure 8. 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

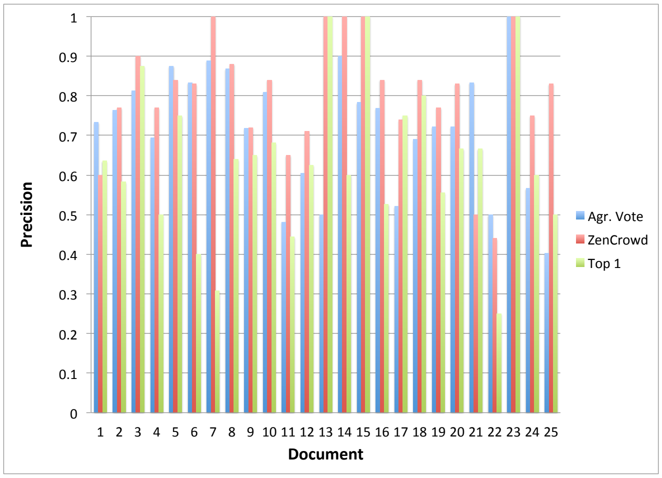

Figure 8: Comparison of three matching techniques.

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 200ms to extract an entity from text, 500ms to select and rank candidate URIs, and 500ms to generate a micro-matching task. The decision process takes on average 100ms. Without taking into account any parallelization, our system can thus offer a new entity to the crowd roughly every second, which in our opinion is sufficient for most applications (e.g., enriching newspaper articles or internal company documents). 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 entity 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).

5.3 Discussion

Looking back at the experimental results presented above, we first observe that crowdsourcing entity matching is useful to improve the effectiveness of an entity linking system. State of the art crowdsourcing techniques can improve precision by 6%. ZenCrowd takes advantage of a probabilistic framework for making decisions and performs even better, leading to performance improvement ranging between 4% and 35% over the manually optimized agreement vote approach, and on average of 14% over our best automatic matching approach. In both cases, the improvement is statistically significant (t-test $p < 0.05$).

A more general observation is that entity linking is a challenging task, which can rapidly become impossible when errors are made at the entity extraction or candidate selection phases. Analyzing the population of workers on the crowdsourcing platform (see Figure 9 left), 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 $[0, 1]$. 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), obtaining perfect linking results is thus in general unrealistic for non-controlled settings. As another consequence, augmenting the numbers of workers performing a given task is not always beneficial: Figure 9—right shows how the average precision of ZenCrowd 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.

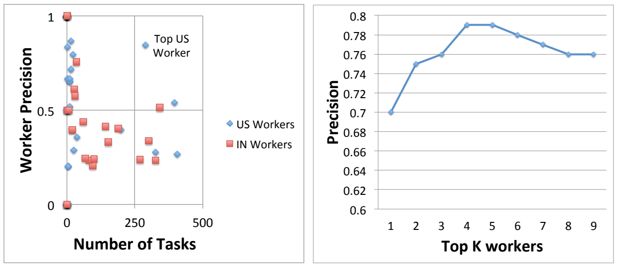

Figure 9: Distribution of worker precision and task precision with top k workers.

6. CONCLUSIONS

As the LOD movement gains momentum, 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 automatic techniques or on the sole work of human experts. In this paper, we have presented ZenCrowd, a system based on a probabilistic framework leveraging both automatic techniques and punctual human intelligence feedback captured on a crowdsourcing platform. ZenCrowd can be used in combination with automatic entity extraction, ranking, and matching techniques to improve the overall linking accuracy.

As our approach incorporates a human intelligence component, it typically cannot perform 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 entity matching in crowdsourcing environments. In conclusion, ZenCrowd provides a reliable approach to entity linking, which exploits the trade-off between *large-scale* automatic entity linking and *high-quality* human annotations, and which according to our results improves the precision of the results by 4% to 35% over a state of the art and manually optimized crowdsourcing approach, and on average by 14% over our best automatic matching approach. As future work, we plan to focus on making the entire linking process more effective and efficient by adopting methods to automatically select the best LOD data sets for a given entity dynamically instead of considering a fixed dataset for all cases. We also plan to focus on a qualitative analysis of our approach by looking into its implications on the end-user side. Finally, we plan to investigate a few sociological aspects by analyzing the long-term behavior of the human workers.

7. ACKNOWLEDGMENT

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

8. REFERENCES

[1] O. Alonso and R. A. Baeza-Yates. Design and Implementation of Relevance Assessments Using Crowdsourcing. In *ECIR*, pages 153–164, 2011.

[2] P. Bailey, A. P. de Vries, N. Craswell, and I. Soboroff. Overview of the TREC 2007 Enterprise Track. In *TREC*, 2007.

[3] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2010 Entity Track. In *TREC*, 2010.

[4] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *IJCAI*, pages 2670–2676, 2007.

[5] R. Blanco, H. Halpin, D. Herzig, P. Mika, J. Pound, H. S. Thompson, and D. T. Tran. Repeatable and reliable search system evaluation using crowdsourcing. In *SIGIR*, pages 923–932, 2011.

[6] R. Blanco, P. Mika, and S. Vigna. Effective and Efficient Entity Search in RDF Data. In *International Semantic Web Conference (ISWC)*, pages 83–97, 2011.

[7] P. Bouquet, H. Stoermer, C. Niederee, and A. Mana. Entity Name System: The Backbone of an Open and Scalable Web of Data. In *Proceedings of the IEEE International Conference on Semantic Computing, ICSC 2008*, pages 554–561.

[8] M. Ciaramita and Y. Altun. 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, pages 594–602, Stroudsburg, PA, USA, 2006. ACL.

[9] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic Message Passing in Peer Data Management Systems. In *International Conference on Data Engineering (ICDE)*, 2006.

[10] P. Cudré-Mauroux, P. Haghani, M. Jost, K. Aberer, and H. De Meer. idMesh: graph-based disambiguation of linked data. In *WWW '09*, pages 591–600, New York, NY, USA, 2009. ACM.

[11] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the ACL*, 2002.

[12] G. Demartini, T. Iofciu, and A. P. de Vries. Overview of the INEX 2009 Entity Ranking Track. In *INEX*, pages 254–264, 2009.

[13] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39, 1977.

[14] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96. ACM, 2005.

[15] A. Feng, M. J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, and R. Xin. CrowdDB: Query Processing with the VLDB Crowd. *PVLDB*, 4(11):1387–1390, 2011.

[16] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. 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, pages 80–88, 2010.

[17] K. Haas, P. Mika, P. Tarjan, and R. Blanco. Enhanced results for web search. In *SIGIR*, pages 725–734, 2011.

[18] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[19] G. Kazai. In Search of Quality in Crowdsourcing for Search Engine Evaluation. In *ECIR*, pages 165–176, 2011.

[20] G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In *SIGIR*, pages 205–214, 2011.

[21] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[22] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *WSDM*, pages 441–450, 2010.

[23] F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 2001.

[24] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

[25] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered Sorts and Joins. *PVLDB*, 5(1):13–24, 2011.

[26] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, 2011.

[27] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[28] B. On, N. Koudas, D. Lee, and D. Srivastava. Group linkage. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 496–505. IEEE, 2007.

[29] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, pages 771–780, 2010.

[30] W. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*, 1999.

[31] M. Wylot, J. Pont, M. Wisniewski, and P. Cudré-Mauroux. dipLODocus[RDF] - Short and Long-Tail RDF Analytics for Massive Webs of Data. In *International Semantic Web Conference (ISWC)*, pages 778–793, 2011.