Holistic and Scalable Ontology Alignment for Linked Open Data

Toni Gruetze
Hasso Plattner Institute
Potsdam, Germany
toni.gruetze@hpi.uni-potsdam.de

Christoph Böhm
Hasso Plattner Institute
Potsdam, Germany
christoph.boehm@hpi.uni-potsdam.de

Felix Naumann
Hasso Plattner Institute
Potsdam, Germany
felix.naumann@hpi.uni-potsdam.de

ABSTRACT

The Linked Open Data community continuously releases massive amounts of RDF data that shall be used to easily create applications that incorporate data from different sources. Inter-operability across different sources requires links at instance- and at schema-level, thus connecting entities on the one hand and relating concepts on the other hand. State-of-the-art entity- and ontology-alignment methods produce high quality alignments for two "nicely structured" individual sources, where an identification of relevant and meaningful pairs of ontologies is a precondition. Thus, these methods cannot deal with heterogeneous data from many sources simultaneously, e.g., data from a linked open data web crawl.

To this end we propose Holistic Concept Matching (HCM). HCM aligns thousands of concepts from hundreds of ontologies (from many sources) simultaneously, while maintaining scalability and leveraging the global view on the entire data cloud. We evaluated our approach against the OAEI ontology alignment benchmark as well as on the 2011 Billion Triple Challenge data and present high precision results created in a scalable manner.

1. INTRODUCTION

In 2006 Berners-Lee proposed the Linked Open Data (LOD) design principles[1]. These principles outline the vision of a global data cloud that can be used to build novel applications incorporating information about real-world entities from many sources. He suggests dereferencable HTTP URIs for naming things that return useful information when looked up on the web. Further, he encourages links to other URIs so that one can discover more information about things under consideration. The semantic web community adopted these suggestions and we are now witnessing the growth of a giant global data cloud comprising information form many sources using de-facto standards such as RDF, RDFS, OWL, etc. The pure availability of this data following a set of principles is a big win since, in general, the use of (non-linked) open data requires source-specific approaches to access, query and filter the data of interest. Instead, when published as LOD, one can leverage different sources through common mechanisms such as HTTP and SPARQL. Examples of such LOD applications are showcased on `data.gov` and `data.gov.uk`.

However, the LOD vision includes the connection of data from different sources via links to facilitate an easy integration. As of September 2011, the LOD cloud comprised 295 sources, which all fulfill the basic LOD principles[2]. In contrast, the number and quality of links across these sources are an ongoing issue since their discovery is a major challenge [1]. In this paper, we focus on schema-level links, i.e., ontology alignments across data sources; of the 295 data sources 190 use proprietary vocabulary terms. Out of these 190 sources, only 15 offer mappings to other widely deployed vocabularies; but 159 provide dereferencable URIs for proprietary terms, i.e., descriptive information for these "new terms" are available. Thus, the evolution of the web of vocabulary terms requires to run ontology alignment approaches on (all) pairs of sources without any (or with only few) mappings to other vocabularies.

State-of-the-art ontology matching has been designed to cope with nicely structured and well defined ontologies in order to produce high-quality mappings for one pair of sources from one specific domain at a time. Instead, in the case of data that stems from the web, we need approaches that can (simultaneously) deal with heterogeneous and incomplete vocabulary definitions from many different sources dealing with various topics. Further, the vocabularies from the LOD cloud as a whole allow a holistic view on the web of vocabulary terms and thus to create alignments depending on other alignments and dependencies. Resulting alignment information across many sources can be used for web query answering or the discovery of sources with respect to specific topics. However, it is a major scalability challenge to deal with very many vocabulary terms gathered from the linked data web. Therefore, we tackle one the major challenges in ontology matching, namely the matching at a very large scale [19]. We further add the requirement to be applicable to real-world web data from various origins instead of two specific sources.

In the following section we briefly review state-of-the-art techniques for aligning ontologies and then derive requirements for an approach the deals with heterogeneous vocabulary definitions from the web of data (Sec. 1.1). Next, we outline our general approach for aligning multiple LOD vocabularies (Sec. 1.2), followed by a technical description of respective alignment phases (Sec. 2–4). Along with the technical details of our approach we present measurements to convince the reader of the respective phase's scalability.

[1]`http://www.w3.org/DesignIssues/LinkedData.html`

[2]`http://www4.wiwiss.fu-berlin.de/lodcloud/state/`

LDOW2012 , April 16, 2012, Lyon, France

1.1 State-of-the-art

The matching of data models is an essential task for various areas in the information science, e.g., information integration, peer-to-peer data management, Web service composition, etc. [8]. Thus, a wide range of approaches were published in the area of schema and ontology matching. Current ontology matching approaches often enter the annual Ontology Alignment Evaluation Initiative (OAEI) [7]. The OAEI aims at comparing the performance of different systems by assessing respective strengths and weaknesses. Successful participants are, for instance, [2–4, 12, 15, 17]. Many state-of-the-art approaches are based on the combination of different basic matching techniques and require a parameter optimization for each matching task. This configuration is addressed by different meta matching systems [5, 16, 22]. However, most of the state-of-the-art approaches have not been run on large and heterogeneous ontologies that stem from the LOD cloud.

Nevertheless, the number of ontology matching approaches dedicated to ontologies from the LOD domain has grown recently. Those LOD approaches commonly use only one (or few) basic matching techniques. Often, these techniques utilize special RDF and LOD characteristics to improve the matching result. For instance, Nikolov et al. introduce an approach that utilizes *owl:sameAs* links between data sets in the LOD cloud to derive concept overlaps [18].

However, there are hundreds of ontologies in the Web of Data, which have been created for different use cases but still contain many overlaps worth discovering. Therefore, the ability to perform cross-domain matching is especially important. Jain et al. introduced an approach that utilizes Wikipedia categories for bootstrapping. This way, they can cover knowledge from various domains [13, 14]. Furthermore, the LOD ontologies are commonly very large and contain many instances. Suchanek et al. use instance knowledge to match instance and schema level entities [25].

Recently, Rahm surveyed different approaches to large-scale matching tasks [20]. Besides different basic techniques to reduce runtime and manual configuration effort, he explicitly emphasized holistic schema matching. Approaches of this type process a set of input schemata in one run (like [23]) and infer knowledge from the entirety of schemata [9, 10, 24]. In this paper we we adopt this notion for ontology alignment on a large scale.

The input for web-scale ontology alignments is a set C of concepts that has been gathered from the LOD cloud via web access methods. The concepts in C stem from many different ontologies. For each concept in C, there is at least a URI – its id. Preferably, there are further information, such as a label, a description, or a comment. Additionally, there can be further meta data like structural information. For the processing of these large amounts of heterogeneous ontological information we suggest the following properties for an alignment approach:

- The approach must work with ontologies from **diverse domains**, i.e, the underlying concept matching strategy must be applicable to many domains.
- Due to a large input concept set, the approach must perform **automatic alignment** in **sub-quadratic runtime** (in the number of concepts).
- The approach should **process concept definitions only**. That is, it can neglect instance data due to its immense size. Further, property definitions can be ignored since we found that property definitions and their actual usage differs largely.
- **Ontology structures should not be used** for the actual alignment creation (if available at all). This is for scalability reasons and since the structure across diverse ontology varies largely and is thus not beneficial. However, the approach can exploit structural information, such as subclass relationships, to verify derived alignments and find semantic contradictions.
- The approach must return **equivalence relationships** for vocabulary terms. These relationships can be fuzzy (depending on some parameter) since different ontologies have different granularities, e.g., *umbel:SpacePlatform_Manned* $\sim$ *dbpedia:SpaceStation*.

Please note that others might make other design decisions; however, due to the immense scale and a questionable quality of current general web data we follow these properties.

1.2 Alignment approach overview

The aforementioned properties shall hold for an approach that can be applied to web-scale input data. However, at the same time we target an holistic view on the data to leverage the information at hand as a whole. Therefore, we propose the Holistic Concept Matching approach (HCM). To address the scalability challenge we group the input data by topic and thus create small groups of concepts that can be aligned locally. Since we group by topics, we can still infer relationships holistically, i.e., draw conclusions not only based on a pairwise similarity but additionally based on alignments and dependencies among other topically related members of the same group.

Consider for instance five sources (A, B, C, D, E) and let $\{a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_o, d_1, \ldots, d_p, e_1, \ldots, e_q\}$ be the concepts from these sources. Running a traditional approach on all available pairs of ontologies would result in up to $\binom{5}{2}$ isolated runs, which is computationally infeasible given that alignment approaches often base on complex lexical and structural properties [2, 15, 17]. Also, isolated runs may yield low-quality results, since vocabulary definitions from the web can be incomplete and highly heterogeneous in terms of granularity and structure of the ontology.

With the help of additional topic knowledge for the sources and their entities we are able to group them. For instance, given that the three sources (A, B, C) store media related entities and (D, E) provide information from the life sciences, we examine groups of concepts, e.g., $\{a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_o\}$ and $\{d_1, \ldots, d_p, e_1, \ldots, e_q\}$ – yielding fewer runs of the approach. Note that these groups must not directly relate to the input ontologies: For instance, if $d_1 = human$, then it could also reside in the first group. However, within these groups, we can then run computationally more complex approaches to take an holistic view based on multiple ontologies.

Specifically, our general approach is shown in Fig. 1. Given the data from the LOD cloud, we extract a knowledge representation for all available concepts. This representation can be a simple descriptive string, a feature vector, or a more complex data structure. Next, we apply topical grouping in order to create smaller sets of concepts. Within

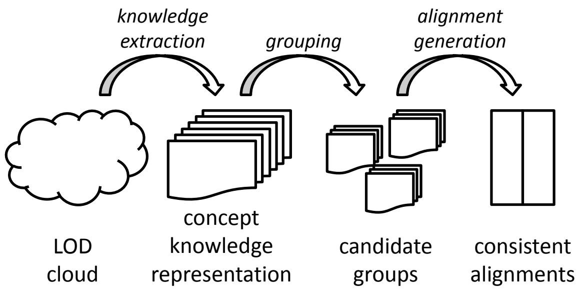

Figure 1: Scalable and holistic LOD ontology alignment

these sets, we then create alignments by identifying similar knowledge representations and reasoning among them using additional structural information from the input as well as further candidates from the same group.

Obviously, many techniques can be plugged into this approach. Depending on the knowledge representation, one can choose the specific representation of a concept's topic, specific topic similarities as well as specific concept similarities in order to find alignments. In the remainder of this paper, we report on the adoption of the Wikipedia category forest [13] for HCM (Sec. 2). The topical grouping is done using a set similarity index (Sec. 3). For the alignment generation we combine the Wikipedia category forest similarity [14] and a rule-based verification approach [15] (Sec. 4).

2. KNOWLEDGE REPRESENTATION

The comparison of different concepts requires an abstract representation of its semantic content, i.e., a knowledge representation. First we present our method to compute this representation, then we show our performance results.

2.1 Wikipedia Category Forests

Given a set C of concepts from many different ontologies gathered from the LOD cloud, we now elaborate on the data structure we use to represent a concept. To this end, we chose Wikipedia Category Forests (WCF) as proposed for the BLOOMS algorithm in [13, 14], as BLOOMS is a state-of-the-art alignment algorithm for matching heterogeneous ontologies from many domains. A WCF is a set of Wikipedia category trees created as follows:

1. Given a single concept $c \in C$, create a list of keywords $kw(c) = \{k_1, \dots, k_n\}$ from c.

2. Given $kw(c) = \{k_1, \dots, k_n\}$, a Wikipedia search for all keywords returns a ranked list of Wikipedia pages $R = \{p_1, \dots, p_m\}$. These pages are roots of the trees in the resulting forest.

3. For each page p and a height parameter h, construct a tree for the h top-ranked pages $\{p_1, \dots, p_h\}$ in R, in the following recursive manner:

   (a) In recursion $1 \leq h$ determine p's categories $\{\alpha_1, \dots, \alpha_q\}$.

   (b) In recursion $2 \leq h$ determine α's super-categories $\{\beta_1, \dots, \beta_r\}$.

   (c) In recursion $3 \leq h$ determine β's super-categories $\{\gamma_1, \dots, \gamma_s\}$.

   (d) etc.

We now explain the three steps on more detail.

Step 1. To determine keywords $kw(c)$ for a concept c we have tested several methods. In the following, we use two orthogonal base methods, namely the *TFIDF*$_n$-extraktor and *ConceptID*-extraktor. The *TFIDF*$_n$-extractor processes descriptions of concepts, i.e., comments and labels. For this, we merge description texts and tokenize them and neglect stop words[3]. Then, we determine the tf-idf-score for each token with respect to the overall corpus of concept descriptions. Finally, we select the top-n tf-idf-ranked tokens as keywords $kw(c)$.

The *ConceptID*-extractor is solely based on the concept's URI. The concept id is the unique identifier of a concept in its ontology definition. HCM uses a straight-forward approach to determine the concept id: We truncate the prefix of the URI until the last occurrence of either #, :, or /. The concept id must not contain any other character than letters or underscores. To extract tokens, we split the concept id at camel-case characters and underscores. After again removing stop words, this results in the set of keywords $kw(c)$. For instance, the concept id for `http://umbel.org/umbel/rc/SpacePlatform_Manned` would be *SpacePlatform_Manned*. From this we create $kw(c) = \{space, platform, manned\}$.

Additionally, HCM supports the *ConceptID_TFIDF*$_n$-extractor that combines both methods. First, ConceptID keywords are extracted. If no WCF can be constructed (see next steps) HCM tries to build a forest using TFIDF$_n$ keywords instead. Thus, the ConceptID_TFIDF$_n$-extractor maximizes the amount of concepts that can be represented by an WCF.

Step 2. Next, HCM performs a Wikipedia full-text search using a search query that is built by concatenating all keywords $kw(c)$ (delimited by spaces). From the resulting Wikipedia pages we choose the top d pages as tree root for the creation of a WCF. The parameter d – the forest depth – influences a WCF's coverage of conceptual meanings. To catch all potential meanings of a concept, several Wikipedia articles and their category hierarchies have to be considered, i.e., it requires a deep forest. On the other hand, a deeper forest has a higher risk to contain trees not related to the actual concept. Therefore, the selection of the forest depth parameter is a trade-off between semantic coverage and irrelevance.

Step 3. The tree construction builds trees recursively from the root to the maximal tree height h. The height mainly influences the level of abstractness of a WCF. The higher the trees, the more category hierarchy layers are considered. This thus leads to more abstract categories in the WCF. Instead, the lower the trees, the less probable is a topical overlap among related concepts. We will discuss different parameter configuration in the experiments section.

For instance given resource *umbel:MannedSpacecraft*, the ConceptID-extractor yields $\{manned, spacecraft\}$, which

[3]using the English stop word lists of Apache Lucene (v2.3.0) and of the MediaWiki-plugin LuceneSearch (v2.1.3) with 33 respectively 121 words

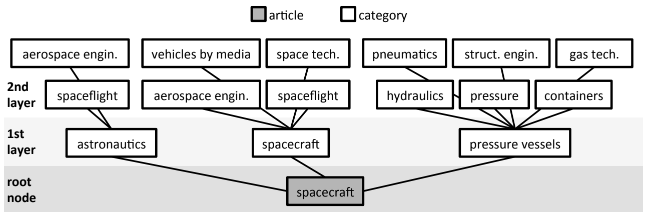

Figure 2: Tree for the Wikipedia article "Spacecraft" with $h = 2$. The 2nd layer categories span two rows for readability.

results in the following Wikipedia search result ($d = 3$): $\{Human\_spaceflight, Spacecraft, Orion\_(spacecraft)\}$. Figure 2 depicts the tree for the root article *Spacecraft* ($h = 2$). As one can see, the higher the layer, the more nodes (Wikipedia categories) exist and the more abstract are these categories. Furthermore, we note that different tree nodes occur multiple times, e.g., "spaceflight" or "aerospace engineering". These nodes play an essential role in the next phase of HCM (Sec. 3).

2.2 Experiments

Setup. All experiments were performed on a Windows 2008 R2 Enterprise Server with two quad-core Intel Xeon processors (2.66 GHz) and 30GB of memory. To evaluate the performance of our approach in a web scale scenario, we used the 2011 Billion Triple Challenge (BTC) data[4]. The BTC data is a crawl from the LOD cloud and consists of approximately 2.2 billion triples. We have implemented all described methods in Java.

The extraction of concept information can be done in linear time with Hdrs – a scalable distributed RDF store[5] – that allows fast in-order index scans. HCM selects concepts by considering resources occurring in triples with different predicates: *rdf:type*, *rdfs:subClassOf*, *rdfs:domain*, *rdfs:range*, *owl:equivalentClass*, *owl:disjointWith*, *rdfs:label*, and *rdfs:comment*. In this manner HCM identified approx. 1M concepts in the BTC.

In contrast to the original BLOOMS algorithm [13], we cannot use the Wikipedia search service. This is because we need to issue very many search requests since we deal thousands of concepts. This vast amount of queries would lead to a high network overhead when querying the search service. Therefore, for HCM, we load a Wikipedia dump[6] into a full-text index (articles and category hierarchy). The index was created with *Apache Lucene*[7] (v2.3.0) and a modified version of the *LuceneSearch*[8] MediaWiki-plugin implementation (v2.1.3). Additionally, we used MongoDB[9] (v1.6.5) running on the same machine like the HCM implementation. HCM uses MongoDB to store intermediate results between the three phases (Sec. 2, 3, and 4).

[4]http://km.aifb.kit.edu/projects/btc-2011

[5]http://code.google.com/p/hdrs

[6]We used the English Wikipedia dump of August 3rd, 2011 with approx. 11 million pages: http://dumps.wikimedia.org/enwiki/20110803/

[7]http://lucene.apache.org/java/docs/index.html

[8]http://www.mediawiki.org/wiki/Extension:Lucene-search

[9]http://www.mongodb.org/

Measurements. Given the BTC data where we could extract 1M concepts, the overall keyword extraction time is 7 minutes (Step 1). The average number of keywords per set, i.e., per concept, is 2.77.

As expected, the Wikipedia index query time (Step 2) is linear to the number of extracted keyword sets. For instance, given 293k keyword sets from the ConceptID_TFIDF$_3$-extractor, querying took 64 minutes.

The input to the keyword set extractors comprises 1M concepts. However, the ConceptID-extractor cannot yield a result for all input concepts, since we do not deal with blank nodes[10] and malformed URIs. Further, not all keyword sets created yield a query result; in the end 238k keyword sets could be used to build WCFs. Nevertheless, when additionally applying the TFIDF$_3$-extractor we can further create another 55k WCFs.

The forest construction runtime mainly depends on forest depth d and tree height h. Table 1 shows runtimes for the forest construction algorithm with different parameters. Apparently, the runtime linearly depends on the forest depth d. An increase of the tree height h leads to an exponential runtime increase.

$h \setminus d$	**5**	**10**	**15**	**20**
1	0:06h	0:09h	0:11h	0:14h
2	0:15h	0:23h	0:28h	0:38h
3	0:31h	0:52h	1:18h	1:34h
4	1:13h	**3:03h**	3:55h	5:45h
5	4:14h	8:28h	14:01h	15:40h

Table 1: Forest construction runtime using different tree h and d parameters and the ConceptID_TFIDF$_{10}$-extractor (in hours)

Similar to Jain et al., we set tree height $h = 4$ and forest depth $d = 10$ for all following experiments [13]. For this configuration, the WCFs consists of 9.53 trees on average.

3. CANDIDATE GROUP CREATION

The previous step extracts WCFs, i.e., our knowledge representation of choice. Given these knowledge representations, we now need to find topically related concepts in order to group them. Note that in the following we use the terms *domain* and *topic* interchangeably. A domain or topic is a field of the real world where different concepts together play a role. The underlying intuition for the following is that a search for alignments among concepts within (and not across) topics should not cause a major loss of recall. Again, we first present our method, followed by an evaluation.

3.1 Topical groups

In our approach, the grouping (by topic) shall reduce the runtime complexity of the alignment generation step by splitting the problem space into smaller independent tasks (Sec. 4). This procedure is often referred to as blocking or partitioning.

Next, we discuss the details of finding topically related WCFs from the previous step. Given a set F of WCFs, we need to identify disjoint groups $G_1, \ldots, G_n \subset F$ of topically

[10]Approx. 50% of all input concept IDs are blank nodes

related forests (topic group). In HCM we implemented the following procedure:

1. For each WCF $f \in F$, extract $\mathit{topic}(f) = \{t_1, \ldots, t_m\}$ from f. We describe a topic of a WCF using a set of tree nodes extracted from the WCF.

2. Given a topic $\mathit{topic}(f_1)$, identify all forests $f_2 \in F$ with a high topical overlap ($\mathit{topic}(f_1) \simeq \mathit{topic}(f_2)$).

3. Given a set of pairs (f_1, f_2) with a high topical overlap, determine groups G of forests by associating topically overlapping forests transitively.

Step 1. The topic $\mathit{topic}(f)$ of a WCF f is a subset of its tree nodes ($\{p, \alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_r, \ldots\}$). In the process of finding characteristic nodes that represent this concept's topic, we ignore very specific categories like "Soviet manned space program" and very generic categories like "Humans" or "Wikipedia article lists". In this paper we report on a simple yet promising approach we have tested (among others) for the topic extraction: The $\mathit{TFIDFForest}_n$-extractor again utilizes the tf-idf measure. Nodes common in a WCF are scored with a high term-frequency-value (tf), whereas nodes popular in many WCFs are scored with a low inverse-document-frequency-value (idf). We rank a WCF's tree nodes in descending tf-idf order and select the top-n for representing the topic. The choice of n is crucial for the quality of the tree node set representing the concepts's topics. Experiments indicated $n = 10$ to be a reasonable choice. Higher values lead to very specific topics, whereas smaller n lead to a low representativity.

Step 2. Next, we compare the topic sets to identify related forests. We use the Jaccard coefficient J and a corresponding threshold θ_J to determine the similarity of topic sets ($J(\mathit{topic}(f_1), \mathit{topic}(f_2)) \geq \theta_J$). The naïve pairwise comparison of all forest topic sets leads to a quadratic runtime behavior with respect to the number of forests $|F|$. To reduce this effort we tested different set-based similarity self join techniques and selected *ppjoin* [27], which delivered the most promising results (see Sec. 3.2). The ppjoin is a set similarity self join technique that reduces the amount of object pairs to be compared by using an inverted index and two filtering techniques on the candidate sets (prefix and positional filtering). Our implementation is an adapted version of an open source implementation[11]. The application of a set-based similarity self join technique enhances performance and still keeps the topical cohesion of the group.

Step 3. To determine groups of topically related WCFs, we build a graph containing forest nodes F and edges between topically related forests ($J(\mathit{topic}(f_1), \mathit{topic}(f_2)) \geq \theta_J$). Next, we use a depth-first search to determine connected components within the graph [11]. Then, all forests in a component are either directly or indirectly connected to each other, but have no connection to forests in other components.

3.2 Experiments

Figure 3 depicts the WCF group distribution in the BTC data over the similarity threshold θ_J. For one, we illustrate

[11]`https://code.google.com/p/similarity-join-tools/`

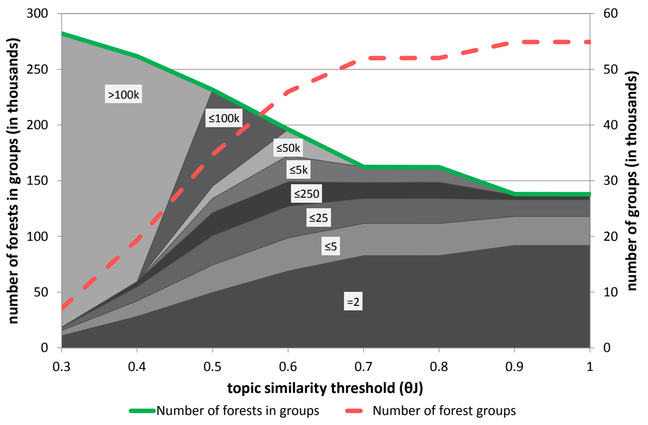

Figure 3: The forest group distribution for $h = 4$, $d = 10$, the ConceptID_TFIDF keyword extractor, and the TFIDFForest$_{10}$ topic extractor for varying θ_J. The green graph shows the number of forests that can be grouped (left axis). The shaded areas below indicate the number of forest in groups of a specific size. The dashed red line shows the total number of groups (right axis).

the number of forests that can be grouped for a given θ_J (green graph, left axis). Shaded areas below the green graph depict the number of forests in groups of a given size, e.g., $\mathit{size} = 2$, $\mathit{size} \leq 5$, $\mathit{size} \leq 25$, etc. Also, we show the total number of groups over θ_J (dashed red graph, right axis).

As for the fraction of WCFs that can be grouped, a higher θ_J leads to a decrease since the higher θ_J, the fewer WCF pairs have a sufficient overlap to be grouped. The shaded areas further show that the higher θ_J, the more WCFs fall in small groups, since a stricter overlap criterion induces more individual groups. Vice versa, a lower θ_J results in larger groups since smaller groups are merged in this case.

As for the total number of groups for a specific Jaccard threshold, there is a clear increase for higher θ_J. This is due to the fact that there are more smaller groups for higher θ_J. The figure shows that $\theta_J = 1.0$ leads to 138k (out of 293k) forests have at least one correspondence with an identical topic set. The majority (92k) appears in groups of size two.

For the following experiments we set $\theta_J = 0.7$ as default for three reasons: (1) This avoids very large WCF groups which would raise the runtime of the alignment generation phase. With $\theta_J = 0.7$, the maximal group size of 4k leads to a reasonable (and feasible) runtime. (2) $\theta_J = 0.7$ provides a significant topical overlap among forests and minimizes haphazard correspondences. (3) This threshold value enables the identification of groups for more than 55% of the available WCFs (162k). Remaining WCFs do not have a sufficient topic overlap and can thus not be considered for the alignment generation.

In general, the runtime of the candidate group generation depends on the number of input WCFs, the applied self-join technique, and topic set homogeneity (within the input set). Figure 4 compares the runtime of three different set similarity join techniques with groups for TFIDFForest$_{10}$ topics, $\theta_J = 0.7$, and varying number of WCFs. The red graph (triangle markers) shows the quadratic runtime of the naïve baseline approach that compares all forests in a pair-

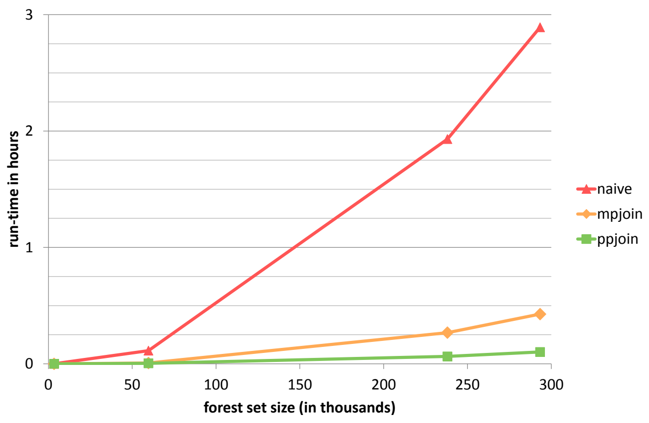

Figure 4: The forest grouping run-time for different similarity join algorithms. The experiment was executed for $\theta_J = 0.7$ using the TFIDFForest$_{10}$ topic extractor and $h = 4$ and $d = 10$.

wise manner. The yellow line (rhomb markers) indicates the performance of the mpjoin introduced by Ribeiro and Härder [21]. In contrast to the ppjoin, mpjoin tries to minimize the effort for the candidate set determination instead of minimizing the candidate set size. The green line (rectangle markers) illustrates the runtime of the ppjoin algorithm. The ppjoin algorithm performs best and has a nearly linear runtime. Here, the similarity join of 295k forest topics takes only 6 minutes. Remember that these numbers have been created with a very large and heterogeneous sample from the current Web of Data – the BTC. Thus, we can demonstrate a reasonable runtime for current very large web data sets. What is more, Wang et al. introduce a MapReduce implementation of ppjoin [26], which leads to the conclusion that our approach can also be run on even larger future datasets.

4. ALIGNMENT GENERATION

Given the set of candidate groups the next task is to regard each group and find alignments among their respective members independently. This leads to runtime reduction decreased due to the reduced input set size ($|G| \ll |F|$) and the possible parallelization of alignment generation task for different groups In the following, we elaborate on our underlying matching strategy to identify relations among concepts, followed by an experimental evaluation.

4.1 WCF alignments

Given a set G of topically related WCFs, we need to identify a set of alignments A. We propose a procedure that, again, consists of three steps:

1. Extend G by adding related WCFs. These relations originate from the original ontology definitions.
2. Compare all forest pairs $f_1, f_2 \in G$ using a forest overlap score function O. Extract all forest pairs with an overlap value exceeding a given threshold $O(f_1, f_2) \geq \theta_O$; add them to match candidate set M.
3. Create an alignment graph D by iteratively adding candidate matches M that neither semantically conflict with ontology definitions, nor with other candidate alignments. Finally, extract alignments A from the conflict-free alignment graph D.

Step 1. First, we extend each topic group G with further related WCFs in order to incorporate immutable axioms from the underlying ontologies. Specifically, related WCFs f_2 have an ontology-level relation in common with a WCF $f_1 \in G$, i.e., *owl:equivalentClass* and *owl:disjointWith* relationships. We did not select other relations since we deem these two as most relevant to reveal semantic conflicts of alignments while keeping the group size low (which is a desirable property). In the following, the topic group G refers to the extended topic group.

Step 2. Next, we compare all pairs of WCFs $f_1, f_2 \in G$ to determine the overlap of the forests. HCM uses the tree overlap similarity of BLOOMS+ to compare individual trees [14]. The tree overlap aggregates common nodes between two trees as follows: Given two arbitrary trees t_1, t_2 from different WCFs, the tree overlap is defined as:

$$Overlap(t_1, t_2) = \frac{\log \sum_{n \in (t_1 \cap t_2)} \left(1 + e^{d(n,t_1)^{-1} - 1}\right)}{\log 2|t_1|}$$

Thus, the tree overlap depends on the distance d of common nodes $n \in (t_1 \cap t_2)$ to the root of t. The higher the node depth in t ($d(n, t)$), i.e., the more nodes are between n and the root page p of t, the smaller the influence of the shared node. The depth of the root page is set to $d(p, t) = 1$. For instance, given the tree t_1 with the root $p =$ "spacecraft" (article) of our previous example (see Fig. 2) and an arbitrary tree t_2. Assuming two overlapping nodes $x =$ "astronautics" and $y =$ "aerospace engin." between t_1 and t_2 such that $t_1 \cap t_2 = \{x, y\}$, the resulting depths of the nodes in t are $d(x, t) = 1$ and $d(y, t) = 2$. Therefore, the resulting tree overlap of both trees in t_1 is:

$$Overlap(t_1, t_2) = \frac{\log \left(\left(1 + e^0\right) + \left(1 + e^{-\frac{1}{2}}\right) \right)}{\log(2 \times 14)} \approx 0.385$$

The tree overlap is not symmetric ($Overlap(t_1, t_2) \neq Overlap(t_2, t_1)$). With this asymmetry BLOOMS+ is able to identify parent-child relationships between concepts. The equivalence between concepts is derived if $Overlap(t_1, t_2) = Overlap(t_2, t_1)$. However, we are interested in a similarity value for two WCFs instead of individual trees. Therefore, we compare all pairs of trees of different WCFs and select the best matching tree pair. Given two WCFs f_1 and f_2 ($f_1, f_2 \in G; f_1 \neq f_2$), we define the forest similarity as the maximal harmonic mean for the overlaps of all tree pairs ($\forall t_1 \in f_1, t_2 \in f_2$):

$$O\left(f_1, f_2\right) = \arg \max_{\substack{t_1 \in f_1,\\ t_2 \in f_2}} \left(\frac{2 Overlap(t_1, t_2) Overlap(t_2, t_1)}{Overlap(t_1, t_2) + Overlap(t_2, t_1)} \right)$$

Then, we select relevant matches M by using an overlap threshold $O(f_1, f_2) \geq \theta_O$. The selection of a good threshold is essential for the alignment quality, because a low θ_O-value leads to a high amount of irrelevant candidates and thus a lower precision. The following semantic deduction might not be able to eliminate erroneous candidates. In addition, the run-time complexity in the semantic verification phase grows. A high θ_O-value instead leads to fewer candidates and thus a low recall. In Sec. 4.2 we evaluate the different thresholds and their influence on the alignment quality.

Step 3. Next, HCM performs a logical reasoning to remove conflicting matches from candidate sets and thus enhance the precision. HCM draws conclusions for matches contradicting each other. This is done using the holistic knowledge of all ontologies. Given a ranked set of forest match candidates M, the semantic verification will produce an alignment graph $D = (G, E)$ with a set of WCF nodes G and a set of directed edges $E \subset G \times G$ representing the verified alignments. The edges are annotated with three different attributes:

- The **certainty** of an alignment $cert : G \times G \to \mathbb{R}_0^1$ specifies the confidence of a matching. The higher the value, the more reliable the match.
- The alignment **type** classifies every matching pair in one of five categories $type : G \times G \to \{equi, disj, parent, child, onto\}$. An *equi* edge connects two concepts identified to be equal, whereas a *disj* edge marks two concepts to be disjoint. A *parent* or *child* edge is a directed subset information representing an *rdfs:subClassOf* relationship. An *onto* labeled edge marks two concepts to originate from the same source ontology.
- The alignment **origin** provides information about the edge's origin, $origin : G \times G \to \{def, detect, infer\}$. A *def* edge is an axiom stated in the source ontology. A *detect* alignment pair is a verified concept alignment pair, i.e., $O(f_1, f_2) \geq \theta_O$. An *infer* edge is a inference drawn from other matches with the help of transitive closures.

To populate the graph D, HCM performs the following:

(a) Initialize the alignment graph D with no edges ($E = \emptyset$).

(b) Populate D with ontology constraint edges a ($cert(a) = 1.0$ and $origin(a) = def$) derived from *rdfs:subClassOf*, *owl:equivalentClass*, and *owl:disjointWith* triples from the underlying RDF data.

(c) Add relations a between WCFs originating from the same ontology with $type(a) = onto$, $cert(a) = 1.0$, and $origin(a) = def$. Here, we use the ontology id from a concept URI by removing the ConceptID (see Sec. 2, Step 1).

(d) Finally, match candidates M (Step 2) are selected greedily: We add matches $a = (f_1, f_2)$ from M with decreasing similarity $O(f_1, f_2)$ to E. New edges in E comprise $cert(a) = O(f_1, f_2)$, $type(a) = equi$, and $origin(a) = infer$ as annotations.

When inserting an alignment – ontology constraints or candidate alignments (Step 2) – we perform three steps:

First, we check whether an alignment a contradicts the existing edges in D. To this end, HCM checks for four different types of contradictions: **Conflicting definitions** are alignments in the graph that are incompatible to an new alignment $a = (f_1, f_2)$. A conflicting definition occurs, if there is an edge $e_x = (f_1, f_2); e_x \in E$ such that $type(e_x) \neq type(a)$, $origin(e_x) \neq infer$, or $cert(e_x) \geq cert(a)$. Furthermore,

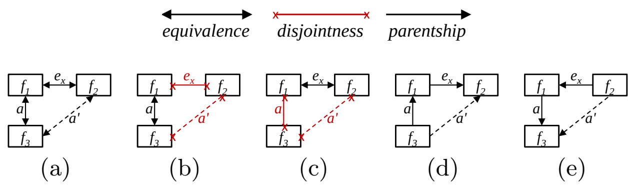

Figure 5: Five types of alignment inferences, that infer a closure a' from a new alignment a and an existing edge e_x in the matching graph. (a) shows an equivalence closure for $type(e_x) = type(a) = equi$. (b) and (c) show disjoint closures for $type(e_x), type(a) \in \{equi, disj\}; type(e_x) \neq type(a)$. (d) and (e) show parentship and childship closures for $type(e_x), type(a) \in \{parent, child\}; type(e_x) = type(a)$.

HCM checks alignments for **multiple-entity correspondences, crisscross correspondences**, and **disjointness-subsumption contradictions**. These three types of contradictions were originally introduced for the ASMOV ontology alignment approach by Jean-Mary et al. [15].

If and only if no contradiction was found for the candidate alignment a, HCM adds the alignment candidate and its inverse a^{-1} to the graph's edge set E. The inverse alignment $a^{-1} = (f_2, f_1)$ of $a = (f_1, f_2)$ has got an analog alignment identification ($origin(a^{-1}) = origin(a)$) and certainty ($cert(a^{-1}) = cert(a)$) and the inverse adapted type ($type(a^{-1}) = type^{-1}(a)$):

$$type^{-1}(x) = \begin{cases} equi & \text{if } type(x) = equi \\ disj & \text{if } type(x) = disj \\ parent & \text{if } type(x) = child \\ child & \text{if } type(x) = parent \\ onto & \text{if } type(x) = onto \end{cases}$$

Second, HCM infers further alignments from a (a^{-1} respectively) by drawing conclusions with the help of existing alignments $e \in E$. HCM supports five types of inference types shown in Figure 5: Given an existing alignments $e = (f_1, f_2)$ $e \in E$ and the alignment just added $a = (f_2, f_3)$ (respectively $a^{-1} = (f_2, f_3)$), a closure $a' = (f_1, f_3)$ can be drawn.

Third, given the verified alignment graphs, HCM outputs all edges $e \in E$ with $origin(e) = detect$ as alignment set A.

For instance, given a WCF group $G = \{dbpedia{:}SpaceStation, umbel{:}SpacePlatform\_Manned, dbpedia{:}Spacecraft\}$ the population of graph D works as follows: The only ontology constraint for G is the alignment $a_{onto} = (dbpedia{:}Spacecraft, dbpedia{:}SpaceStation)$, with $cert(a_{onto}) = 1$, $origin(a_{onto}) = def$, and $type(a_{onto}) = onto$ in order to indicate that both concepts to originate from the same ontology. Both alignments a_{onto} and the inverse a_{onto}^{-1} are added to D. Afterwards the candidate set $G = \{a_1, a_2, a_3\}$ is processed. Assuming alignment certainties of:
$a_1 = (dbpedia{:}SpaceStation, umbel{:}SpacePlatform\_Manned)$
$cert(a_1) = 0.955$,
$a_2 = (dbpedia{:}Spacecraft, dbpedia{:}SpaceStation)$
$cert(a_2) = 0.718$, and
$a_3 = (dbpedia{:}Spacecraft, umbel{:}SpacePlatform\_Manned)$
$cert(a_3) = 0.341$
HCM proceeds as follows: First a_1 is added to D with $origin(a_1) = detect$ and $type(a_1) = equi$ (and so a_1^{-1}). Af-

terwards, a_2 is not added, due to the conflicting definition a_{onto}, which marks both concepts to originate from the same ontology. Finally, due to a multiple-entity correspondence a_3 is prevented too: *umbel:SpacePlatform_Manned* is already matched to a concept from the *dbpedia*-ontology.

4.2 Experiments

In the following section, we discuss the experimental results of HCM for the BTC data as well as on the Ontology Alignment Evaluation Initiative benchmark track [6, 7].

BTC. To determine HCM's alignment precision for the BTC data, we executed the alignment generation algorithm using different threshold settings (θ_O). We chose seven ranges: $sim_{forest} = 1$, $1 > sim_{forest} \geq 0.95$, $0.95 > sim_{forest} \geq 0.9$, ..., $0.75 > sim_{forest} \geq 0.7$. We did not consider alignments with $O(f_1, f_2) < 0.7$ which is along the lines with the argument by Jain et al. for the BLOOMS+'s tree overlap measure [14]. The authors mention an optimal threshold of 0.85. After determining alignments using HCM, a random sample of 50 alignments for each similarity range was selected. Then three independent annotators manually inspected these alignments and identified the actual relation between aligned concepts in a majority vote process. We used three different types of relations: **Equivalence** indicates a correct alignment of two equivalent concepts. **Similar** indicates a fuzzy match. The two concepts are related and the set of instances of the concepts have a significant intersection. **Disjoint** indicates an incorrect alignment of two concepts. The set of instances of the concepts are disjoint.

Table 2 shows our manual evaluation results as well as their confidence intervals (confidence level of 95%). Additionally, it depicts absolute numbers of alignments found in the BTC data for varying thresholds. In the case of $1 > sim_{forest} \geq 0.95$, 57.5% of the alignments were labeled as equivalences and 15% as similar. The number of actual equivalences decreases for smaller forest similarities. When considering only the strict equivalence judgments as relevant alignment, the inter-annotator agreement for strict equivalences is very high $\kappa = 0.909$ (Cohen's kappa coefficient). By treating more fuzzy results (similar or equal alignments) as correct, the inter-annotator agreement decreases to $\kappa = 0.817$. This is due to a varying perception of a similarity or relatedness. It is much more subjective than a strict equivalence definition. For instance, the concepts *daml:Ammeters* and *umbel:Voltmeter* are related due to their common purpose. On the other hand, from a taxonomic point of view, the instance sets of both concepts are probably disjoint. HCM discovered interesting matches such as:

- *yago:PsychoactiveFungi* and *umbel:HallucinogenicMushroom*
- *dbpedia:BirdsOfAustralia* and *opencyc:Parrot*
- *umbel:AirConditioner* and *dbpedia:HeatExchangers*

Note that, due to diverse and non-trivial domains in the BTC corpus, these kind of matches are hard to identify, even for humans. Our algorithm can deal with these cases because it exploits broad knowledge for many domains present in Wikipedia.

Furthermore, it is interesting to see that even in case of false positives, the algorithm reveals interesting semantic relations between concepts. For instance, HCM aligns the concepts *yago:Outlaws* and *umbel:MotorcycleClub* due to a semantic relation via *outlaw motorcycle clubs* similar to *Rivals Hells Angels* and *Bandidos*.

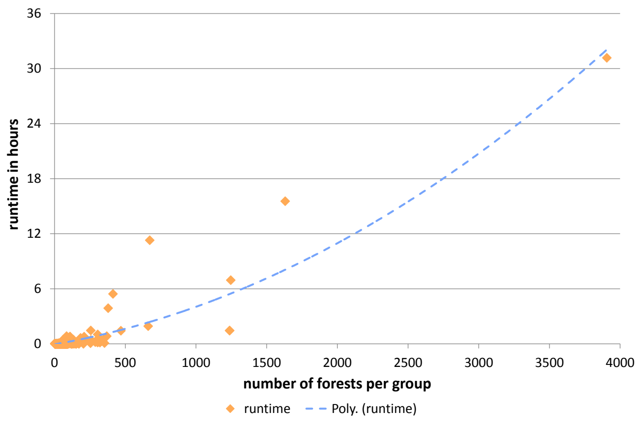

Figure 6: Alignment generation execution time for different WCF group sizes. The blue dotted line indicates the quadratic regression line unifying the actual alignment generation time per group (indicated by a yellow rhomb). The experiment was executed with $h = 4$, $d = 10$, the ConceptID_TFIDF$_3$ keyword extraction method, the TFIDFForest$_{10}$ topic extractor ($\theta_J = 0.7$), and the forest threshold $\theta_O = 0.7$.

Runtimes of the alignment generation phase is visualized in Figure 6 for different WCF groups (orange rhomb marker). Due to the pairwise comparison of trees of all WCFs in a group G to generate candidate alignments (see Step 2), the alignment generation phase has a quadratic runtime in the group size $|G|$. Furthermore, the effort alignment verification is polynomial to the amount of alignment candidates. The blue dotted line indicates the polynomial regression of the forest match execution times. The amount of small groups is very high (see Figure 3), which is no problem, because of small run-times. A remaining issue is the large effort of the matching step for large groups which is left for future work.

$O(f_1, f_2)$	$P(eq)$	$P(eq \cup sim)$	align.
$O = 1.0$	0.964 ±0.036	0.964 ±0.036	601
$1.0 > O \geq 0.95$	0.575 ±0.143	0.725 ±0.129	133.317
$0.95 > O \geq 0.9$	0.481 ±0.145	0.706 ±0.131	72.131
$0.9 > O \geq 0.85$	0.071 ±0.066	0.294 ±0.131	6.921
$0.85 > O \geq 0.8$	0.053 ±0.053	0.200 ±0.114	4.279
$0.8 > O \geq 0.75$	0.053 ±0.053	0.126 ±0.092	3.139
$0.75 > O \geq 0.7$	0.071 ±0.066	0.331 ±0.136	1.950

Table 2: The probabilities and confidence intervals for the 50 random samples of the alignments proposed by HCM (confidence level 95%). The leftmost column shows the different similarity classes ($O(f_1, f_2)$). The 2nd column from the left presents the probability and confidence interval for an alignment representing an equivalence. The 3rd column from the left shows the probability and confidence interval for an alignment representing either equivalence or similarity. The rightmost column depicts the total number of identified alignments within the BTC data.

OAEI. In order to compare with other ontology alignment approaches (e.g. [2, 13, 15, 17]), we provide results on the benchmark track of the Ontology Alignment Evaluation Initiative Campaign [6]. The goal of the track is to evaluate strengths and weaknesses of different alignment approaches by providing a set of alignments between a reference and systematically defamiliarized ontologies. All considered ontologies in the benchmark track originate from the bibliographic domain. Since our algorithm is designed to match concepts from various ontologies at once, we generate alignments for concepts from ontologies in the benchmark track. To determine the matching quality we extract all concept pairs between ontologies having a reference alignment. HCM treats concepts with equivalent URI as one entity. Therefore we implicitly infer alignments between concepts with an equivalent URIs. Furthermore, the reference alignments for properties are neglected, because HCM does not target property alignments. Using a WCF overlap threshold of $\theta_O = 0.7$ leads to a precision of 85% and a recall of 55%. In comparison to other approaches that additionally offered alignments between properties, these numbers are average. This is mainly due to the changes of ConceptIDs and descriptions in the automatically generated ontologies. For instance, the concept *xsqlknk* of ontology *266* neithers contain a human readable ConceptID, nor a description. Nevertheless, the reference alignments contain a match with *101:MastersThesis.* Many state-of-the-art ontology matching algorithms solve this problem by using an structure based similarity as sole basis of decision making. However, for the LOD ontology matching task, this approach is unpromising, because different scopes of the authors of LOD data sets lead to large differences in the ontology structure. For instance, the two data sets of DBpedia[12] and YAGO[13] represent knowledge extracted from Wikipedia. Nevertheless, the ontologies of both data sets vary significantly in structure, whereas many common concepts are shared.

In 2011 a new bibliographic benchmark data set was generated [7]; the results of HCM are the equivalent for the new data set.

5. CONCLUSION AND FUTURE WORK

In this paper, we tackle the problem of *large-scale* concept matching – one of the major challenges in ontology matching [19]. In contrast to other approaches that process pairs of ontologies, our focus is on aligning many ontologies from the LOD cloud simultaneously in a holistic manner. To this end, we propose an abstract workflow (knowledge extraction, grouping, and alignment generation) to specifically enable scalability while still examining all ontologies in its entirety. Further, we plug state-of-the-art techniques and novel ideas into this workflow and report on promising results for scalability and alignment quality in a web-scale alignment scenario. For representing knowledge, we chose Wikipedia Category Forests. For grouping the input, we leverage topical information. Last, the alignment generation leverages Wikipedia Category Forest overlaps and performs a semantic inspection.

We have many ideas for future directions: For instance, we will look into other knowledge representations and matching techniques that can be plugged into the workflow. The challenge here, is to identify approaches that can capture different semantic notions of an input concept and allow a useful grouping. Also, we plan to derive further relationship types - maybe with the help of instance data instead of Wikipedia categories. Last, we plan to experiment with other topical groupings that, e.g., allow overlaps across groups. Another grouping method could also support incremental updates and thus facilitate online ontology alignments for the growing web of linked data.

[12]`http://dbpedia.org/`
[13]`http://www.mpi-inf.mpg.de/yago-naga/yago/`

References

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 4(2):1–22, 2009.

[2] I. F. Cruz, F. P. Antonelli, and C. Stroe. AgreementMaker: Efficient matching for large real-world schemas and ontologies. In *Proceedings of the VLDB Endowment*, pages 1586–1589, 2009.

[3] J. David, F. Guillet, and H. Briand. Matching directories and OWL ontologies with AROMA. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 830–831, 2006.

[4] H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002.

[5] K. Eckert, C. Meilicke, and H. Stuckenschmidt. Improving ontology matching using meta-level learning. In *ESWC*, pages 158–172, 2009.

[6] J. Euzenat, A. Ferrara, C. Meilicke, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb Zamazal, V. Svátek, and C. Trojahn. Results of the Ontology Alignment Evaluation Initiative 2010. In *Proceedings of the International Workshop on Ontology Matching at ISWC*, 2010.

[7] J. Euzenat, A. Ferrara, W. R. van Hage, L. Hollink, C. Meilicke, A. Nikolov, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb Zamazal, and C. Trojahn. Final results of the Ontology Alignment Evaluation Initiative 2011. In *Proceedings of the International Workshop on Ontology Matching at ISWC*, 2011.

[8] J. Euzenat and P. Shvaiko. *Ontology Matching.* Springer-Verlag, 2007.

[9] B. He and K. C.-C. Chang. Statistical Schema Matching across Web Query Interfaces. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 217–228, 2003.

[10] B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *Proceedings of the Conference on Knowledge Discovery and Data Mining*, pages 148–157, 2004.

[11] J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378, 1973.

[12] J. Huber, T. Sztyler, J. Nößner, and C. Meilicke. Codi: Combinatorial optimization for data integration: results for oaei 2011. In *Proceedings of the International Workshop on Ontology Matching (OM)*, 2011.

[13] P. Jain, P. Hitzler, A. P. Sheth, K. Verma, and P. Z. Yeh. Ontology Alignment for Linked Open Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2010.

[14] P. Jain, P. Z. Yeh, K. Verma, R. G. Vasquez, M. Damova, P. Hitzler, and A. P. Sheth. Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, 2010.

[15] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7:235–251, 2009.

[16] Y. Lee, M. Sayyadian, A. Doan, and A. S. Rosenthal. etuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal*, 16:97–122, January 2007.

[17] J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21:1218–1232, 2008.

[18] A. Nikolov, V. Uren, E. Motta, and A. de Roeck. Overcoming Schema Heterogeneity between Linked Semantic Repositories to Improve Coreference Resolution. In *Proceedings of the Asian Conference on The Semantic Web*, pages 332–346. Springer Berlin / Heidelberg, 2009.

[19] S. Pavel and J. Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012, to appear.

[20] E. Rahm. Towards Large-Scale Schema and Ontology Matching. In *Schema Matching and Mapping*, chapter 1, pages 3–27. Springer Berlin / Heidelberg, 2011.

[21] L. A. Ribeiro and T. Härder. Efficient Set Similarity Joins Using Min-prefixes. In *Advances in Databases and Information Systems*, pages 88–102. Springer Berlin / Heidelberg, 2009.

[22] D. Ritze and H. Paulheim. Towards an Automatic Parameterization of Ontology Matching Tools based on Example Mappings. In P. Shvaiko, J. Euzenat, T. Heath, C. Quix, M. Mao, and I. Cruz, editors, *Proceedings of the 6th International Workshop on Ontology Matching*, volume 814, pages 37–48, `http://ceur-ws.org`, October 2011. CEUR Workshop Proceedings.

[23] K. Saleem, Z. Bellahsene, and E. Hunt. PORSCHE: Performance ORiented SCHEma mediation. *Information Systems*, 33:637–657, 2008.

[24] W. Su, J. Wang, and F. Lochovsky. Holistic Schema Matching for Web Query Interfaces. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 77–94, 2006.

[25] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, 5:157–168, 2011.

[26] C. Wang, J. Wang, X. Lin, W. Wang, H. Wang, H. Li, W. Tian, J. Xu, and R. Li. MapDupReducer: detecting near duplicates over massive datasets. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1119–1122, 2010.

[27] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 131–140, 2008.