Ontology Instance Linking: Towards Interlinked Knowledge Graphs

Jeff Heflin
Department of Computer Science and Engineering
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015, USA

Dezhao Song
Research and Development
Thomson Reuters
610 Opperman Drive
Eagan, MN 55123, USA

Abstract

Due to the decentralized nature of the Semantic Web, the same real-world entity may be described in various data sources with different ontologies and assigned syntactically distinct identifiers. In order to facilitate data utilization and consumption in the Semantic Web, without compromising the freedom of people to publish their data, one critical problem is to appropriately interlink such heterogeneous data. This interlinking process is sometimes referred to as *Entity Coreference*, i.e., finding which identifiers refer to the same real-world entity. In this paper, we first summarize state-of-the-art algorithms in detecting such coreference relationships between ontology instances. We then discuss various techniques in scaling entity coreference to large-scale datasets. Finally, we present well-adopted evaluation datasets and metrics, and compare the performance of the state-of-the-art algorithms on such datasets.

Introduction

Recently, more and more data is being published in Semantic Web formats (e.g., the Resource Description Framework (RDF)[1] and the Web Ontology Language (OWL)[2]) by both academic researchers and industrial organizations. The New York Times (NYT) has published data in Semantic Web format, currently consisting of 5,000 people, 1,500 organizations and 2,000 locations[3]. The British Broadcasting Corporation (BBC) has also published their data in RDF, covering a much more diverse collection of entities[4], e.g., persons, places, events, etc. Thomson Reuters, an intelligent information and news provider, also provides free access to an RDF version of its data[5] (3.5 million companies, 1.2 million equity quotes, and so on).

Using natural language processing and information extraction techniques, academics have also created DBpedia (Lehmann et al. 2015), an RDFized Wikipedia, that describes about 3.7 million entities with a total of 400 million facts. Various efforts are also being investigated for building and improving a multilingual DBpedia. Freebase (Bollacker et al. 2008), similar to DBpedia but with a much larger amount of 1.7 billion facts, covers entities of various types, ranging from entertainment, sports to arts, musics, books and to medicine and biology.

Linked Data (Bizer, Heath, and Berners-Lee 2009) is a term used to describe best practices for publishing and connecting data using RDF and Uniform Resource Identifiers (URIs). There are currently about 1,000 datasets in the Linked Open Data (LOD) cloud, amounting to 74 billion triples and 250 million links across different datasets[6]. These data are often independently generated and distributedly stored in many locations, and are also heterogeneous and covering diverse domains, including People, Geographic, Publications, Media, Social Web, etc.

Although the amount of Semantic Web data is rapidly growing in many different domains, one observation is that each real-world entity (e.g., people, organizations, songs, books, etc.) in the Semantic Web may be described and published by many data publishers with syntactically distinct identifiers. For example, CiteSeer and DBLP assign different identifiers to *Tim Berners-Lee*, the inventor of the World Wide Web, and describe him in distinct ways with complementary information. Importantly, such identifiers from different data sources are often not linked to each other and thus prevent end users from easily obtaining relatively comprehensive information for the entities.

In order to help users to better utilize and consume Semantic Web data, in recent years, there has been a great amount of research efforts in trying to interlink ontology instances from different datasets, i.e., generating equivalence linkages between ontology instances. In the Semantic Web, the *owl:sameAs* predicate is used to connect equivalent ontology instances. Such *owl:sameAs* links help to build an interlinked Semantic Web and enable users to more easily obtain relatively comprehensive descriptions of the entities.

This interlinking problem has been studied by Natural Language Processing researchers as the *Entity Coreference* or *Entity Resolution* problems (Bagga and Baldwin 1998), and by Database researchers as the *Deduplication* or *Record Linkage* problems (Elmagarmid, Ipeirotis, and Verykios 2007). In the Semantic Web, an ontology instance (e.g., person, location, book, etc.) is identified with a URI while syntactically distinct URIs could actually represent the

[1] http://www.w3.org/TR/rdf11-primer
[2] http://www.w3.org/TR/owl-ref/
[3] http://data.nytimes.com/
[4] http://www.bbc.co.uk/things/
[5] https://permid.org/
[6] http://stats.lod2.eu

same real-world entity. We will use the term *Entity Coreference* to refer to the process of finding ontology instances that refer to the same real-world entity throughout this paper.

Producing high quality equivalence relationships for the current Semantic Web is a non-trivial task. First of all, a coreference algorithm should be able to generate accurate and comprehensive equivalence links. Various factors can impact the coreference results, e.g., misspellings, missing information, etc. Furthermore, scalability issues need to be taken into account. Although it might be possible to perform manual linking on small datasets, automatic and particularly scalable approaches will be needed to detect equivalence linkages across large-scale heterogeneous datasets. Many Linked Data datasets have millions of instances (e.g., DBpedia (Lehmann et al. 2015) and Freebase (Bollacker et al. 2008)), therefore brute-force approaches that carefully compare every pair of instances will not scale well.

Finally, domain-independence should be another desired property of coreference algorithms. A domain refers to the category (e.g., People, Geographic, Publications, Media, etc.) and the usage (e.g., academic people, politics, etc.) of the data. In the past, domain-specific techniques have been able to achieve good coreference results, e.g., by relying on common name variations to identify coreferent person instances; frameworks have also been designed for manually specifying rules for linking ontology instances, e.g., Silk (Volz et al. 2009). However, when considering various domains, humans may lack the knowledge or time to specify what information to utilize and thus many interesting domains may end up unlinked. Given the diverse domains covered by Linked Data, coreference algorithms that work well across different domains are desired.

In this paper, we will discuss the various efforts in the area of interlinking ontology instances. We will first categorize and discuss the different approaches in generating *owl:sameAs* links. Furthermore, we will discuss datasets and benchmarks that have been adopted for evaluating such entity coreference systems. We will then present and compare the results of the state-of-the-art systems. Finally, we will discuss the potential future directions of entity coreference in the Semantic Web.

Approaches

Entity coreference has drawn interests from researchers in a variety of fields. For free text, a key task is to decide which name mentions actually represent the same real-world entity (Bagga and Baldwin 1998). In databases, entity coreference is better known as record linkage or deduplication and is used to detect duplicate database records (Elmagarmid, Ipeirotis, and Verykios 2007). In the Semantic Web, entity coreference can happen between a free text mention and an ontology instance (Hassell, Aleman-Meza, and Arpinar 2006; Mendes et al. 2011) or between ontology instances themselves. The latter has received more attention from the research community, since being able to automatically provide high quality *owl:sameAs* links between heterogeneous and large-scale datasets is recognized as one critical step toward building an interlinked data web.

Linking Ontology Instances with String Matching

Various string matching-based approaches have been proposed. *LogMap* (Jiménez-Ruiz and Grau 2011) computes the similarity between the "labels" of two ontology instances and picks the highest similarity between any pair of labels of the two instances as their final similarity score. Here, "label" is broadly defined but limited to objects of datatype properties (i.e., properties whose values are strings). One potential drawback would be that using the highest similarity score between values of any manually determined property pairs as the final similarity measure for two instances could result in too many false positives, because two non-coreferent instances might coincidently share highly similar values for their "labels", e.g., two people with the name "John Smith" may not necessarily be coreferent.

RiMOM (Wang et al. 2010) combines manually specified property weights with string matching techniques for detecting coreference relationships. The core idea is that different properties may be more or less informative and thus for each property, a specific weight is assigned. However, when there are a large number of predicates (e.g., the Billion Triples Challenge dataset[7]), manually determining property weights could be a really time-consuming process or even not feasible. Differently, *SERIMI* (Araújo et al. 2015) and *EPWNG* (Song and Heflin 2013) automatically compute a discriminability value for each property, i.e., the degree to which the values for the property are different across instances, capturing the importance of each property for coreference.

Combing Logical Reasoning with String Matching

Instead of only relying on matching literal values, logic based approaches have also been proposed. ObjectCoref (Hu, Chen, and Qu 2011) adopts a two-step approach for detecting coreferent instances. First, it builds an initial set of coreferent instances via reasoning, i.e., by using the formal semantics of OWL properties, such as *owl:sameAs*, *owl:InverseFunctionalProperty* and *owl:FunctionalProperty*. In a second step, it learns the discriminability of property pairs based on the initially discovered coreferent instance pairs. The discriminability reflects how well each pair of properties can be used to determine whether two instances are coreferent or not. Similarly, LN2R (Saïs, Pernelle, and Rousset 2009), CODI (Noessner et al. 2010) and ASMOV (Jean-Mary, Shironoshita, and Kabuka 2009) also combine reasoning and string matching techniques.

One disadvantage of reasoning based approaches is that they highly depend on the correct expressions of the ontologies. For example, as reported by ASMOV researchers, in one dataset, the *surname* property was declared to be functional, however it is possible that a person may have two surnames: one prior to marriage and one after. Another potential weakness of logic-based approaches is that they may not be applicable to non-Semantic Web data, since there are no formal semantics. For instance, for relational databases and XML/CSV data, we do not have the properties listed

[7] http://km.aifb.kit.edu/projects/btc-2012/

above (e.g., *owl:FunctionalProperty*), thus limiting the benefits of the logic layer. Planning-based approaches are also proposed to determine the optimal linking rules for link discovery (Ngomo 2014).

Ontology Instance Matching with Crowdsourcing

The approaches discussed above are all automatic in the sense that except for having to manually specify the weights of different types of triples in some algorithms, the coreference results are achieved by an automatic system without human intervention. However, during the past years, several algorithms that consider human involvement for improving coreference results have been proposed (Cheng, Xu, and Qu 2015; Demartini, Difallah, and Cudré-Mauroux 2013). Typically, automatically generated results are published on some crowdsourcing platforms as evaluation tasks, such as Amazon Mechanical Turk[8] and CrowdFlower[9], and humans can then provide their judgments on the tasks. Their responses can then be aggregated, combined with similarity scores or utilized for active learning (Isele and Bizer 2013).

One potential risk of using crowdsourcing is that many of the evaluators are simply doing the tasks for money and are often times not spending sufficient time to really understand the tasks. This will then cause noisy results. To alleviate this issue, some crowdsourcing services (e.g., Amazon Mechanical Turk) try to identify high-quality evaluators. Also, researchers themselves may also intersperse tasks with known answers to identify reliable evaluators.

Scaling Entity Coreference Systems

Scalability has become an important issue for entity coreference systems. Blocking is one method for subdividing entities into mutually exclusive blocks and only those within the same block will be compared. Instead of finding mutually exclusive blocks, blocking is also referred to as finding a set of candidate pairs of mentions that could be coreferent (Michelson and Knoblock 2006; Song and Heflin 2011).

Manually Identifying Blocking Key Domain expertise has been widely adopted for blocking. Best Five (Winkler 2005) is a set of manually identified rules for matching census data. Sorted Neighborhood (SN) (Hernández and Stolfo 1995) sorts all entities on one or more key values and compares identifiers in a fixed-sized window. Adaptive Sorted Neighborhood (ASN) (Yan et al. 2007) sorts records based upon a manually identified key and learns dynamically sized blocks for each record. The authors claimed that changing to different keys didn't affect the results but provided no details on how they reached this conclusion.

Although keys selected by domain experts can be very effective in many scenarios, the required expertise may not be available for various domains. Moreover, even when people have the required expertise, they may lack the time to actually write down the rules.

Automatic Blocking Key Selection *BSL* (Michelson and Knoblock 2006) adopted supervised learning to learn a blocking scheme: a disjunction of conjunctions of (method, attribute) pairs. For example, a "method" could be "computing the Jaccard similarity between two attribute values". It learns one conjunction each time to reduce as many pairs as possible; by running the learning process iteratively, more conjunctions would be obtained in order to increase coverage on true matches. However, supervised approaches require sufficient training data that may not always be available. For example, when *BSL* was used on the Restaurant dataset[10], a reduction of training set data by 80% led to a 4.68% reduction in true matches identified. Even more importantly, *BSL* may not scale well to large datasets, since essentially it needs to try out every possible combination of (method, attribute) pairs and picks the best one (in terms of pair reduction and coverage of true matches) at each learning iteration. In order to reduce the needs of training data, Cao et. al. (2011) proposed a similar algorithm that utilizes both labeled and unlabeled data for learning the blocking scheme; however the supervised nature of their method still requires a certain amount of available groundtruth.

Differently, Adaptive Filtering (Gu and Baxter 2004) is unsupervised and it filters record pairs by computing their character bigram similarity. Marlin (Bilenko and Mooney 2003) uses an unnormalized Jaccard similarity on the tokens between attributes by setting a threshold to 1, i.e., finding an identical token between the attributes. Although it was able to cover all true matches on some datasets, it only reduced the pairs to consider by 55.35%. If we apply this approach to datasets with millions of instances ($> 10^{12}$ candidate pairs), this reduction is unlikely to be significant enough.

DisNGram (Song and Heflin 2011) learns the blocking key for a selected set of entity categories by iteratively combining the top-key (ranked automatically by a metric) with other attributes. Differently, *TYPiMatch* (Ma and Tran 2013) learns blocking keys for each specific subtype. For example, instead of learning a general key for the class "Person", it learns separate keys for subtypes, such as "Student" and "Professor". *KD2R* (Pernelle, Saïs, and Symeonidou 2013) and *SAKey* (Symeonidou et al. 2014) are two other unsupervised methods for discovering keys in the Semantic Web. Different from *BSL* (Michelson and Knoblock 2006), both systems discover blocking keys without having to consider all possible combinations of the properties.

Index-based Blocking Inverted indexes are commonly used for blocking or finding similar strings. In general, these approaches first index the input values to look up initial candidates, which are then further refined to produce the final candidate pairs. *PPJoin+* (Xiao et al. 2011), *RiMOM* (Wang et al. 2010), and *DisNGram* (Song and Heflin 2011) index on tokens while *LogMap* (Jiménez-Ruiz and Grau 2011) also indexes their lexical variations (e.g., using WordNet); differently, *EdJoin* (Xiao, Wang, and Lin 2008) and *IndexChunk* (Qin et al. 2011) index on character n-grams. In order to reduce index size, *PPJoin+*, *EdJoin*, and *IndexChunk* only index a prefix of a given string and *DisNGram* automati-

[8] https://www.mturk.com/mturk/welcome

[9] http://crowdflower.com/

[10] http://www.cs.utexas.edu/users/ml/riddle/data.html

cally selects the attributes whose entire values are indexed. Rather than only indexing the strings, Ioannou et al. (2010) builds an inverted index by hashing a neighborhood RDF graph of an ontology instance. Furthermore, *PPJoin+* and *IndexChunk* consider the position of the matching tokens/n-grams for filtering while *EdJoin* also takes mismatching n-grams into account. Instead of performing exact matching, FastJoin (Wang, Li, and Feng 2014) adopts fuzzy matching by combing token and character-based similarity. In addition to inverted indices, other types of indices have also been adopted, e.g., B-Tree and Trie (Jiang et al. 2014).

Evaluating Ontology Instance Matching

Here, we will introduce the well-adopted metrics and datasets for evaluating ontology instance matching systems.

Evaluation Metrics

Three metrics have been well adopted for evaluating ontology instance matching systems (Euzenat et al. 2010; Hu, Chen, and Qu 2011; Song and Heflin 2013): Precision (P_t), Recall (R_t) and F1-score ($F1_t$) as computed in Equation 1. Precision is measured as the number of correctly detected coreferent pairs divided by the total number of detected pairs; Recall is defined as the number of correctly detected coreferent pairs divided by the total number of coreferent pairs given a set of ontology instances; F1-score gives a comprehensive view of how well a system performs:

$$P_t = \frac{|correctly\ detected|}{|all\ detected|}, R_t = \frac{|correctly\ detected|}{|true\ matches|}, \quad F1_t = 2 * \frac{P_t * R_t}{P_t + R_t} \tag{1}$$

where t represents threshold in all the above three equations.

To evaluate blocking techniques, three traditional metrics have been frequently used (Yan et al. 2007; Michelson and Knoblock 2006; Song and Heflin 2011; Ngomo 2012): Pairwise Completeness (PC), Reduction Ratio (RR) and $F1\text{-}score_{cs}$ (F_{cs}) as shown in Equation 2. PC and RR evaluate how many true positives are retained by a blocking algorithm and the degree to which it reduces the number of pairwise comparisons needed respectively; F_{cs} is the F1-score of PC and RR:

$$PC = \frac{|true\ matches\ in\ candidate\ set|}{|true\ matches|}$$
$$RR = 1 - \frac{|candidate\ set|}{N * M}, F_{cs} = 2 * \frac{PC * RR}{PC + RR} \tag{2}$$

where N and M are the sizes of two instance sets that are matched to one another. Blocking techniques should have a high PC so that most of the true matches (coreferent instance pairs) will be included in the candidate set. Additionally, a high RR is also important, since a blocking algorithm also needs to be able to reduce as many instance pairs as possible to save the overall computational cost.

Note that according to the definition of *RR*, a large change in the size of the candidate set may only be reflected by a small change in the *RR* due to its large denominator. Therefore, there is the need to adopt new evaluation methods and metrics to perform a more fair comparison between different systems on very large datasets. One option would be to apply an actual coreference algorithm to the selected candidate pairs to: 1) measure the runtime of both blocking and entity coreference; 2) check how the missing true matches can affect the final coreference results. It is possible that even if those missing pairs were selected, the coreference algorithm would still not be able to detect them. Furthermore, in order to cover the last few missing true matches, more false positives could be selected, which would potentially add more computational complexity to the entire process.

Evaluation Datasets

First of all, the Ontology Alignment Evaluation Initiative (OAEI)[11] includes an instance matching track from 2009 that provides several benchmark datasets. On one hand, synthetic datasets are provided each year, which are generated by modifying one data source according to various criteria, e.g., data transformation and deletion. On the other hand, some real-world tasks (e.g., linking the New York Times data to DBpedia) are also provided. Furthermore, in our prior work, two datasets: RKB[12] and SWAT[13], were adopted; both datasets contain millions of instances, which is reasonable to demonstrate the scalability of linking algorithms.

Although the OAEI benchmark has been well-adopted, the main issues are the size of the provided datasets and their number of limited domains. In general, only a few thousand ontology instances are involved in a benchmark dataset, which may not be ideal for evaluating the scalability of coreference algorithms. Also, one common issue of OAEI, RKB, and SWAT is that the data is often of limited domains (e.g., people, locations, organizations, etc.) and thus may not be broad enough to demonstrate the domain-independence of a coreference algorithm.

Finally, we introduce the Billion Triples Challenge (BTC) dataset. The BTC datasets were crawled from the web using a few seeded URIs. Take the BTC2012 dataset as an example (Table 1). First of all, there are 57K predicates and

Number of Triples	1.4 Billion
Number of Instances	183 Million
Number of Predicates	57,000
Number of Classes	296,000

Table 1: Billion Triples Challenge 2012 Dataset Statistics

296K classes in this dataset, thus making the BTC dataset appropriate for testing the domain-independence of an entity coreference algorithm. Furthermore, given the amount of the instances in the dataset, it provides a perfect testbed to study the scalability of coreference algorithms. Neither manually linking nor brute-force approaches will work at this scale.

When adopting real-word datasets for evaluation, due to the low quality of existing *owl:sameAs* links in Linked Data (Halpin et al. 2010), it would be necessary to check the quality of the groundtruth, e.g., by performing manual verification on a sample of the provided groundtruth. Also, since it could be very difficult to obtain perfect

[11] http://oaei.ontologymatching.org/

[12] http://www.rkbexplorer.com/data/

[13] http://swat.cse.lehigh.edu/resources/data/

groundtruth for large real-world datasets, the metric *Relative Recall* ($relR$) may be adopted to compare different systems: $\frac{correctly\ detected\ ontology\ instance\ pairs\ from\ one\ system}{correctly\ detected\ ontology\ instance\ pairs\ from\ all\ systems}$.

In addition to datasets, a few benchmarking tools have also been proposed (Saveta et al. 2015; Ferrara et al. 2011). In general, such tools take an ontology instance as input and perform certain modifications to generate a coreferent instance. Such modifications may include value transformation (e.g., token deletion and stemming), structural transformation (e.g., merging the values of two attributes into one), and semantic transformation (e.g., replacing a property with an equivalent one: "first_name" and "given_name"). Such benchmarking tools are typically configurable and can be used to generate large-scale datasets as well.

Evaluation Results

First of all, in Table 2, we compare the state-of-the-art algorithms on the Person-Restaurant datasets from OAEI (Euzenat et al. 2010). Person1 and Person2 are two synthetic datasets where coreferent records are generated by modifying the original records; Restaurant is a real-world dataset, matching instances describing restaurants from Fodors (331 instances) to Zagat (533 instances) with 112 duplicates.

In general, the majority of the compared systems achieve perfect results on Person1, while we observe significant performance drop for many of the systems on Person2. This is due to how coreferent instances were generated in these two datasets. For Person1, a coreferent instance is created by making only one modification to the original instance; while for Person2, a maximum of 3 modifications per attribute and a maximum of total 10 modifications for all attribute values are allowed. Rather than only using the immediate triples of an ontology instance, several of the best-performing systems, including *EPWNG*, *SiGMa* and *MA*, also utilize values whose distance from the instance in the RDF graph is greater than one. Such distant triples are particularly helpful when there are not sufficient immediate literal triples. Although *SiGMa* and *MA* generally outperform the other systems on all three datasets, both systems assume one-to-one mappings between instances of two datasets, which may not hold in many scenarios.

In Table 3, we also compare the state-of-the-art blocking algorithms on 100K instances from RKB and SWAT. Generally, *DisNGram* was able to achieve better results than the other systems, particularly for the overall runtime. This is primarily due to the fact that *DisNGram* only performs blocking on automatically selected attribute values; combined with effective character n-gram pruning, it was able to avoid producing too many candidate instance pairs.

Furthermore, Table 4 shows the actual coreference results on 50K and 100K randomly selected instances from BTC. The BTC dataset covers more domains and thus is a perfect testbed for testing the domain-independence of coreference algorithms. We compare *SERIMI* (Araújo et al. 2015), *LogMap* (Jiménez-Ruiz and Grau 2011), *EPWNG* (Song and Heflin 2013) and their variations. For *LogMap_DisNGram*, we replaced *LogMap*'s blocking module with the *DisNGram* algorithm (Song and Heflin 2011). *EPWNG_EdJoin* and *EPWNG_DisNGram* utilize *EdJoin* (Xiao, Wang, and Lin 2008) and *DisNGram* for blocking respectively; both systems adopt the *EPWNG* algorithm for the actual coreference.

Dataset	System	P	R	$F1$
Person1	LN2R (Saïs, Pernelle, and Rousset 2009)	**1.00**	**1.00**	**1.00**
	ASMOV (Jean-Mary, Shironoshita, and Kabuka 2009)	**1.00**	**1.00**	**1.00**
	CODI (Noessner et al. 2010)	0.87	0.96	0.91
	RiMOM (Wang et al. 2010)	**1.00**	**1.00**	**1.00**
	ObjectCoref (Hu, Chen, and Qu 2011)	**1.00**	0.99	0.99
	PARIS (Suchanek, Abiteboul, and Senellart 2011)	**1.00**	**1.00**	**1.00**
	SiGMa (Lacoste-Julien et al. 2013)	**1.00**	**1.00**	**1.00**
	EPWNG (Song and Heflin 2013)	**1.00**	**1.00**	**1.00**
	MA (Xue and Wang 2015)	**1.00**	**1.00**	**1.00**
Person2	LN2R (Saïs, Pernelle, and Rousset 2009)	0.99	0.88	0.93
	ASMOV (Jean-Mary, Shironoshita, and Kabuka 2009)	0.70	0.24	0.35
	CODI (Noessner et al. 2010)	0.83	0.22	0.36
	RiMOM (Wang et al. 2010)	0.95	0.99	0.97
	ObjectCoref (Hu, Chen, and Qu 2011)	**1.00**	0.90	0.95
	SiGMa (Lacoste-Julien et al. 2013)	**1.00**	**1.00**	**1.00**
	EPWNG (Song and Heflin 2013)	0.99	0.99	0.99
	MA (Xue and Wang 2015)	**1.00**	**1.00**	**1.00**
Restaurant	LN2R (Saïs, Pernelle, and Rousset 2009)	0.76	0.75	0.75
	ASMOV (Jean-Mary, Shironoshita, and Kabuka 2009)	0.70	0.70	0.70
	CODI (Noessner et al. 2010)	0.71	0.72	0.72
	RiMOM (Wang et al. 2010)	0.86	0.77	0.81
	ObjectCoref (Hu, Chen, and Qu 2011)	0.58	**1.00**	0.73
	PARIS (Suchanek, Abiteboul, and Senellart 2011)	0.95	0.88	0.91
	SiGMa (Lacoste-Julien et al. 2013)	**0.98**	0.96	0.97
	EPWNG (Song and Heflin 2013)	0.75	0.99	0.85
	MA (Xue and Wang 2015)	N/A	N/A	**0.98**

Table 2: Performance Comparison on the OAEI PR Datasets. P: Precision; R: Recall; $F1$: F1-score between P and R

Dataset	System	F_{cs}	$F1$	T
RKB Person	DisNGram (Song and Heflin 2011)	**0.9968**	**0.9306**	**9.98**
	FastJoin (Wang, Li, and Feng 2014)	0.9940	0.9239	21.86
	PPJoin+ (Xiao et al. 2011)	0.9939	0.9240	29.68
	EdJoin (Xiao, Wang, and Lin 2008)	0.9957	0.9283	20.90
SWAT Person	DisNGram (Song and Heflin 2011)	0.9932	0.9490	**9.03**
	FastJoin (Wang, Li, and Feng 2014)	0.9954	**0.9499**	23.56
	PPJoin+ (Xiao et al. 2011)	0.9952	**0.9499**	21.93
	EdJoin (Xiao, Wang, and Lin 2008)	**0.9959**	0.9494	19.46
RKB Publication	DisNGram (Song and Heflin 2011)	**0.9999**	**0.9974**	**20.76**
	FastJoin (Wang, Li, and Feng 2014)	0.9971	0.9969	308.25
	PPJoin+ (Xiao et al. 2011)	0.9967	0.9966	239.14
	EdJoin (Xiao, Wang, and Lin 2008)	0.9974	0.9966	54.76

Table 3: Blocking and Coreference Results on RKB and SWAT. F_{cs}: the F1-score between PC and RR; $F1$: final coreference F1-score; T (seconds): total runtime

System	50K		100K	
	F1	T	F1	T
LogMap (Jiménez-Ruiz and Grau 2011)	0.33	1,496	N/A	N/A
SERIMI (Araújo et al. 2015)	0.59	37,639	N/A	N/A
LogMap_DisNGram	0.76	990	0.75	2,104
EPWNG_EdJoin	0.68	2,019	0.68	3,971
EPWNG_DisNGram	**0.88**	2,074	**0.88**	2,960

Table 4: Final Coreference Results on BTC. $F1$: Final Coreference F1-score; T (seconds): Overall runtime of both blocking and coreference

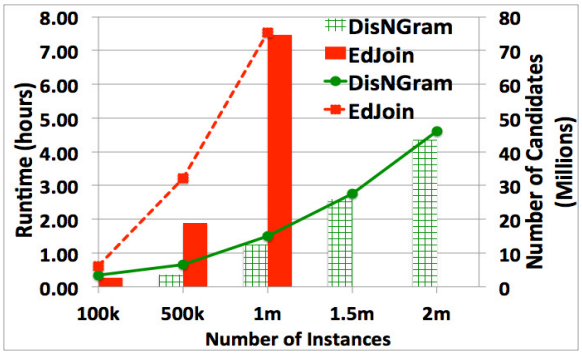

Figure 1: BTC Blocking Scalability. Bars are candidate size; Lines are runtime

Finally, in Figure 1, we compare *DisNGram* and *EdJoin* (the best blocking systems from Table 3 in terms of F_{cs} and T) on up to 2 million instances from BTC. Here, *DisNGram* selects fewer candidates and runs faster than *EdJoin* does.

Conclusion

The volume of Semantic Web data is rapidly growing in a variety of domains. However, unless this data is integrated together, we will not be able to exploit and demonstrate its real value. In this short survey, we summarize the numerous research efforts in recent years that have been devoted to this ontology instance matching problem with the goal of building an interlinked Semantic Web.

In order to handle large-scale datasets, blocking techniques have drawn great interests from the community. To learn the blocking keys (a set of attributes), approaches that do not have to consider all combinations of attributes are preferred. Next, indexes are often created on the values of the keys for look-up, and such look-up results are generally further refined with string matching on the token or character level to produce the final candidates. Through evaluation, some of the discussed algorithms have demonstrated orders of magnitude faster processing capability than other state-of-the-art systems on large datasets. Such speedup is crucial for integrating large-scale datasets in the Big Data era.

As for the actual entity coreference, state-of-the-art systems often times apply weight to different attributes, trying to capture their importance in differentiating the ontology instances. Furthermore, rather than only using immediate triples of the ontology instances, a neighborhood graph that consists of distant triples of the instances is also considered in several of the best-performing algorithms.

In future work, collective entity coreference may be one interesting idea. Rather than detecting coreferent instances of each individual domain, one might imagine how would the coreference results of one type of instances impact the others. For academic datasets, suppose we first find coreferent publications, could we then use this information to improve coreference of authors, especially those that do not have discriminative names? Also, could we automatically determine which domains should be processed first so that the other domains could benefit most? Although similar ideas have been proposed before (Bhattacharya and Getoor 2007), more efforts are needed to generalize such approaches to various domains.

Furthermore, more research is needed to handle data that lacks discriminative labels (e.g., people names and publication titles). One preliminary idea is to combine values from multiple properties, expecting the combined values to be more discriminating than that of any individual property. Moreover, instead of performing exact index look-up, fuzzy matching could be explored. Also, architecturally, distributed computing could be employed to speed up both the blocking and the overall coreference process.

References

Araújo, S.; Tran, D. T.; de Vries, A. P.; and Schwabe, D. 2015. SERIMI: class-based matching for instance matching across heterogeneous datasets. *IEEE Trans. Knowl. Data Eng.* 27(5):1397–1410.

Bagga, A., and Baldwin, B. 1998. Entity-based cross-document coreferencing using the vector space model. In *COLING-ACL*, 79–85.

Bhattacharya, I., and Getoor, L. 2007. Collective entity resolution in relational data. *TKDD* 1(1).

Bilenko, M., and Mooney, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Ninth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, 39–48.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.* 5(3):1–22.

Bollacker, K. D.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD Int'l Conference on Management of Data*, 1247–1250.

Cao, Y.; Chen, Z.; Zhu, J.; Yue, P.; Lin, C.-Y.; and Yu, Y. 2011. Leveraging unlabeled data to scale blocking for record linkage. In *22nd Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 2211–2217.

Cheng, G.; Xu, D.; and Qu, Y. 2015. Summarizing entity descriptions for effective and efficient human-centered entity linking. In *Int'l Conference on World Wide Web*, 184–194.

Demartini, G.; Difallah, D. E.; and Cudré-Mauroux, P. 2013. Large-scale linked data integration using probabilistic reasoning and crowdsourcing. *VLDB J.* 22(5):665–687.

Elmagarmid, A. K.; Ipeirotis, P. G.; and Verykios, V. S. 2007. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.* 19(1):1–16.

Euzenat, J.; Ferrara, A.; Meilicke, C.; Nikolov, A.; Pane, J.; Scharffe, F.; Shvaiko, P.; Stuckenschmidt, H.; Svb-Zamazal, O.; Svtek, V.; and Trojahn dos Santos, C. 2010. Results of the ontology alignment evaluation initiative 2010. In *4th Int'l Workshop on Ontology Matching (OM)*.

Ferrara, A.; Montanelli, S.; Noessner, J.; and Stuckenschmidt, H. 2011. Benchmarking matching applications on the semantic web. In *8th Extended Semantic Web Conference*, 108–122.

Gu, L., and Baxter, R. A. 2004. Adaptive filtering for efficient record linkage. In *SIAM Int'l Conference on Data Mining*.

Halpin, H.; Hayes, P. J.; McCusker, J. P.; McGuinness, D. L.; and Thompson, H. S. 2010. When owl: sameas isn't the same: An analysis of identity in linked data. In *9th Int'l Semantic Web Conference (ISWC)*, 305–320.

Hassell, J.; Aleman-Meza, B.; and Arpinar, I. B. 2006. Ontology-driven automatic entity disambiguation in unstructured text. In *Int'l Semantic Web Conference*, 44–57.

Hernández, M. A., and Stolfo, S. J. 1995. The merge/purge problem for large databases. In *ACM SIGMOD Int'l Conference on Management of Data*, 127–138.

Hu, W.; Chen, J.; and Qu, Y. 2011. A self-training approach for resolving object coreference on the semantic web. In *The 20th Int'l Conference on World Wide Web*, 87–96.

Ioannou, E.; Papapetrou, O.; Skoutas, D.; and Nejdl, W. 2010. Efficient semantic-aware detection of near duplicate resources. In *Extended Semantic Web Conference*, 136–150.

Isele, R., and Bizer, C. 2013. Active learning of expressive linkage rules using genetic programming. *J. Web Sem.* 23:2–15.

Jean-Mary, Y. R.; Shironoshita, E. P.; and Kabuka, M. R. 2009. Ontology matching with semantic verification. *Journal of Web Semantics* 7:235–251.

Jiang, Y.; Li, G.; Feng, J.; and Li, W. 2014. String similarity joins: An experimental evaluation. *PVLDB* 7(8):625–636.

Jiménez-Ruiz, E., and Grau, B. C. 2011. LogMap: Logic-based and scalable ontology matching. In *10th Int'l Semantic Web Conference*, 273–288.

Lacoste-Julien, S.; Palla, K.; Davies, A.; Kasneci, G.; Graepel, T.; and Ghahramani, Z. 2013. SIGMa: simple greedy matching for aligning large knowledge bases. In *19th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, 572–580.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; and Bizer, C. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* 6(2):167–195.

Ma, Y., and Tran, T. 2013. Typimatch: type-specific unsupervised learning of keys and key values for heterogeneous web data integration. In *Sixth ACM Int'l Conference on Web Search and Data Mining (WSDM)*, 325–334.

Mendes, P. N.; Jakob, M.; García-Silva, A.; and Bizer, C. 2011. Dbpedia spotlight: shedding light on the web of documents. In *7th Int'l Conference on Semantic Systems*, 1–8.

Michelson, M., and Knoblock, C. A. 2006. Learning blocking schemes for record linkage. In *Twenty-First National Conference on Artificial Intelligence*, 440–445.

Ngomo, A. N. 2012. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *11th Int'l Semantic Web Conference*, 378–393.

Ngomo, A. N. 2014. HELIOS - execution optimization for link discovery. In *Int'l Semantic Web Conference*, 17–32.

Noessner, J.; Niepert, M.; Meilicke, C.; and Stuckenschmidt, H. 2010. Leveraging terminological structure for object reconciliation. In *7th Extended Semantic Web Conference (ESWC)*, 334–348.

Pernelle, N.; Saïs, F.; and Symeonidou, D. 2013. An automatic key discovery approach for data linking. *J. Web Sem.* 23:16–30.

Qin, J.; Wang, W.; Lu, Y.; Xiao, C.; and Lin, X. 2011. Efficient exact edit similarity query processing with the asymmetric signature scheme. In *ACM SIGMOD Int'l Conference on Management of Data*, 1033–1044.

Saïs, F.; Pernelle, N.; and Rousset, M.-C. 2009. Combining a logical and a numerical method for data reconciliation. *Journal on Data Semantics XII* 12:66–94.

Saveta, T.; Daskalaki, E.; Flouris, G.; Fundulaki, I.; Herschel, M.; and Ngomo, A.-C. N. 2015. LANCE: Piercing to the heart of instance matching tools. In *14th Int'l Semantic Web Conference*.

Song, D., and Heflin, J. 2011. Automatically generating data linkages using a domain-independent candidate selection approach. In *Int'l Semantic Web Conference*, 649–664.

Song, D., and Heflin, J. 2013. Domain-independent entity coreference for linking ontology instances. *J. Data and Information Quality* 4(2):7.

Suchanek, F. M.; Abiteboul, S.; and Senellart, P. 2011. PARIS: probabilistic alignment of relations, instances, and schema. *PVLDB* 5(3):157–168.

Symeonidou, D.; Armant, V.; Pernelle, N.; and Saïs, F. 2014. SAKey: Scalable almost key discovery in RDF data. In *13th Int'l Semantic Web Conference (ISWC)*, 33–49.

Volz, J.; Bizer, C.; Gaedke, M.; and Kobilarov, G. 2009. Discovering and maintaining links on the web of data. In *Int'l Semantic Web Conference (ISWC)*, 650–665.

Wang, Z.; Zhang, X.; Hou, L.; Zhao, Y.; Li, J.; Qi, Y.; and Tang, J. 2010. Rimom results for OAEI 2010. In *5th Int'l Workshop on Ontology Matching*.

Wang, J.; Li, G.; and Feng, J. 2014. Extending string similarity join to tolerant fuzzy token matching. *ACM Trans. Database Syst.* 39(1):7.

Winkler, W. E. 2005. Approximate string comparator search strategies for very large administrative lists. Technical report, Statistical Research Division, U.S. Census Bureau.

Xiao, C.; Wang, W.; Lin, X.; Yu, J. X.; and Wang, G. 2011. Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.* 36(3):15.

Xiao, C.; Wang, W.; and Lin, X. 2008. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *Proc. VLDB Endow.* 1(1):933–944.

Xue, X., and Wang, Y. 2015. Using memetic algorithm for instance coreference resolution. *Knowledge and Data Engineering, IEEE Transactions on* PP(99):1–1.

Yan, S.; Lee, D.; Kan, M.-Y.; and Giles, C. L. 2007. Adaptive sorted neighborhood methods for efficient record linkage. In *Joint Conference on Digital Libraries*, 185–194.