A Scalable Approach to Incrementally Building Knowledge Graphs

Gleb Gawriljuk[1], Andreas Harth[1], Craig A. Knoblock[2], and Pedro Szekely[2]

[1] Institute of Applied Informatics and Formal Description Methods (AIFB), Karlsruhe Institute of Technology, 76128 Karlsruhe, Germany
[2] Information Sciences Institute, University of Southern California, Marina Del Rey, CA 90292, United States of America

Abstract. We work on converting the metadata of 13 American art museums and archives into Linked Data, to be able to integrate and query the resulting data. While there are many good sources of artist data, no single source covers all artists. We thus address the challenge of building a comprehensive knowledge graph of artists that we can then use to link the data from each of the individual museums. We present a framework to construct and incrementally extend a knowledge graph, describe and evaluate techniques for efficiently building knowledge graphs through the use of the MinHash/LSH algorithm for generating candidate matches, and conduct an evaluation that demonstrates our approach can efficiently and accurately build a knowledge graph about artists.

1 Introduction

To be able to link the data about artists from the 13 museums and archives of the American Art Collaborative[1], we need a reference data set that contains all or most of these artists. Ideally, such a reference set would combine the known information from all of the high-quality sources available today. Given that such a reference set does not exist, we have to create the set from the input sources. Constructing a consolidated set is challenging as we need to link the common entities across large sources. Linking entities is challenging because the data is often inconsistent due to errors, misspellings, or out-of-date information.

We present an approach to building knowledge graphs by consolidating data from multiple sources. Constructing such a knowledge graph involves several challenges. First, when consolidating the data from various data sources, we had to align the data across available sources. Second, given the size of the data sources we had to develop a scalable approach that could link the data across sources with millions of entities. Finally, since the web is subject to a constant growth, we designed an approach that makes it easy to include new sources by extending an existing knowledge graph with the data of any new data sources that become available.

[1] `http://americanartcollaborative.org/`

In our approach, each knowledge graph provides data about entities of a specific type. We define a knowledge graph as a set of typed properties between an entity and its property values. Given multiple datasets, we use an initial dataset to create an initial knowledge graph, and then incrementally consolidate further datasets into the knowledge graph. To consolidate data sources into the knowledge graph, we use a five-step approach. One step in the approach applies the MinHash/LSH (Locality-sensitive Hashing) algorithm [8] to produce candidates for rule-based matching functions. Therefore, we reduce the number of comparisons the matching functions must execute, and thus are able to perform entity linkage at scale.

In the remainder of the paper, we first present a motivating example of building a knowledge graph of artists. Second, we describe our scalable approach to incrementally building knowledge graphs. Third, we cover an experiment in which we construct a knowledge graph consolidating 161,465 artist entities from initially 17,539,125 entities, to demonstrate the scalability of the approach. We evaluate precision and recall on a manually built ground truth with 200 artist entities. Finally, we compare the work to other related work and present our conclusions.

2 Motivating example

To illustrate our objective, we provide an example of consolidating data about the pop artist Roy Lichtenstein from four sources. We use the available SPARQL endpoints to access data from the Union List of Artist Names (ULAN), containing descriptions of 109,415 artists provided by the Getty[2], and the Smithsonian American Art Museum (SAAM), containing descriptions of 8,407 artists. We use the downloadable RDF files from DBpedia, which provides data about 1,176,759 people, and the Virtual International Authority File (VIAF), which provides data about 16,244,546 people[3]. Table 1 shows which data source provides what data about Roy Lichtenstein.

After all four sources are consolidated, the resulting knowledge graph holds the data about Roy Lichtenstein from all four source within one cluster. The cluster has links to the identifiers denoting Roy Lichtenstein from all four consolidated data sources, including provenance information. In case two or more sources provide the same value for a property, the knowledge graph mentions the value once and adds provenance for each source which includes the value.

3 Building and extending a knowledge graph

We first give a general overview, and then discuss each step in detail. Our approach is designed to incrementally consolidate one data source at a time. The

[2] `http://www.getty.edu/`

[3] Please note that not all of the people in DBpedia and VIAF are artists.

Table 1. List of properties expressed in the shared domain ontology by data source.

Property name	ULAN	SAAM	DBpedia	VIAF	Knowledge graph
name	X	X	X	X	X
alternateName		X		X	X
givenName		X	X	X	X
familyName		X	X	X	X
gender	X				X
nationality	X				X
birthDate	X	X	X	X	X
deathDate	X	X	X	X	X
birthPlace	X	X	X		X
deathPlace	X	X	X		X
description	X	X	X	X	X

initial data source is used to create an initial knowledge graph. With each subsequent data source as input, we extend the knowledge graph with data of the subsequent data source as shown in Figure 1.

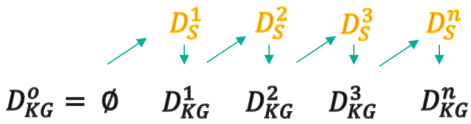

Fig. 1. An illustration of the incremental growth of the knowledge graph.

Our approach consists of five steps as illustrated in Figure 2. In the first step, the input data source is filtered to only select the entities relevant for the final knowledge graph. Then, the schema of the input data source is mapped to either the input knowledge graph or, if no input knowledge graph is provided, to the schema which should be used for the construction of the initial knowledge graph.

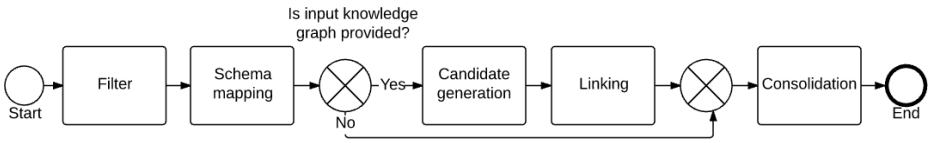

Fig. 2. The distinct steps of the approach.

After the input data source is filtered and mapped, there is a choice. In case no input knowledge graph is provided, the input data source is used to build an initial knowledge graph. For each entity in the initial data source, an entity in the knowledge graph is created in the consolidation step. In case an existing knowledge graph is provided next to the input data source, we continue with

the candidate generation step, in which we apply the MinHash/LSH to generate candidates within the input knowledge graph for entities in the data source. The linking step evaluates each generated candidate using rule-based matching functions to decide which candidate is a match. In the final step, the input data source is consolidated with the input knowledge graph based on the matching links selected by the matching functions.

The approach can be adjusted through parameters concerning the candidate generation and the linking steps. For the candidate generation, the user can specify MinHash/LSH parameters which define how strict the MinHash/LSH searches are for the candidates. For the linking step, the user can define the rule-based matching functions and similarity thresholds.

3.1 Filter

In case the input data source includes additional entity types, the filter step can be applied as an optional step to identify the entities of the target type in the data source. In the filter step, we execute the candidate generation step followed by the linking step for the input data source with the input knowledge graph. By applying the filtering step, we link entities in the data source to the knowledge graph and, thus, identify the linked entities as entities of the target type. Then, the approach proceeds to the next step only with the identified entities.

3.2 Schema mapping

The schema of the input data source has to be mapped to the schema of the knowledge graph to be generated. Both the candidate generation and the linking step require to know which property in the data source semantically corresponds to a property in the knowledge graph. This is defined by the schema mapping. We use Karma [10], which automates the mapping from a source to an ontology, to build the source mappings for the datasets. For mapping the sources, we use `schema:Person` from the schema.org ontology.

3.3 Candidate generation

The most time-consuming task in our approach is linking the entities from the data source to the entities in the knowledge graph. As mentioned by Doan et al. [4] hashing techniques can be applied to scale up the entity linkage. Therefore, to improve the scalability of our approach, we execute the candidate generation step to apply the MinHash/LSH hashing technique [8] before the entity linkage.

We apply the MinHash/LSH algorithm to find candidate entities in the knowledge graph for each entity in the data source which are likely to match, so that we only need to compare the found candidates. Hence, we significantly reduce the number of pair-wise comparisons required to execute and, therefore, speed up the approach.

MinHash/LSH operates over an n-gram representation of the name values[4]. The algorithm then hashes similar entities into the same cluster. The similarity of two entities is defined by the Jaccard similarity between the two sets of n-grams representing the two entities[5].

How well the MinHash/LSH performs regarding recall and precision depends on the number of used minhashes m and the number of items in the generated hashes i. With m and i the LSH threshold t can be approximated as $t = \frac{1}{i}^{\frac{1}{m}}$. Thus, by adjusting m and i we can adjust the LSH threshold.

The LSH threshold t has the characteristic that t is the Jaccard-similarity between the entity e_1 in the knowledge graph and the entity e_2 in the data source so that e_1 has a 50% chance to become a candidate. In other words, if the LSH threshold is set to 80%, an entity e_1 has a 50% chance to be selected as a candidate of entity e_2, when the Jaccard-similarity of e_1 and e_2 is 0.8. Hence, the threshold can be seen as a parameter to adjust the LSH trade-off between recall, precision, and performance. If a high recall is important then the threshold needs to be low and the candidate generation will require more time. If the performance of the MinHash/LSH and precision are important the threshold needs to be high.

To select the most effective LSH threshold for finding candidate based on the entity's name, we conduct extensive experiments of the MinHash/LSH. We assess the MinHash/LSH with either alternative m or alternative i which results in various LSH thresholds. Table 2 shows the resulting LSH thresholds with their recall and precision on the ground truth described in Section 4.1.

Table 2. LSH thresholds with their recall and precision.

t	34%	40%	46%	52%
Precision	14.12%	15.69%	23.12%	26.96%
Recall	100%	99.50%	99.50%	98.01%

The objective of the candidate generation step is to find as many matching candidates as possible. Hence, we need to keep the recall high. To have a high recall, we apply the MinHash/LSH with a low threshold of 46%. A low threshold leads to a low precision because we find many false candidates. However, we tolerate a low precision of the candidate generation because the precision will be increased in the following linking step.

3.4 Linking

The linking step evaluates the clusters of candidates resulting from the MinHash/LSH algorithm. We apply rule-based matching functions to decide whether

[4] The 2-gram of the first name 'Roy' consists of { _R, Ro, oy, y_ }.

[5] The Jaccard similarity between sets S and T is defined as $\frac{|S \cap T|}{|S \cup T|}$.

the candidate and the target entity in the knowledge graph are the same real-world entity. The objective for the matching functions is to increase the precision as much as possible while not decreasing the recall.

When applying a matching function, the function iterates over each cluster. For each cluster, the function examines its matching rules for each candidate. If the candidate fulfills the matching rules, the candidate stays in the cluster; otherwise, the candidate is removed. As the clusters contain the data of each candidate and the target entity, the matching rules can leverage that data to decide whether a candidate is a match or not.

The matching functions can be exchanged with each other or chained to a sequence of functions, each evaluating a certain aspect of the candidate data. For example, when comparing people entities, we can define two matching functions to first check the similarity of the names and remove candidate with a low similarity and, then, further remove candidates with a different birth year.

3.5 Consolidation

In the final consolidation step, we either construct an initial knowledge graph from the initial data source or extend the input knowledge graph with the selected links of the input data source.

The data model of the knowledge graph illustrated in Figure 3 is based on the `schema:Role` concept, which provides a way to represent multiple provenance information for one property value. Each distinct property value of an entity is modeled as a feature role holding the property value and its provenance. We use the properties of `schema:Person` to describe artists and the PROV-O[6] ontology to model the provenance properties. The provenance indicates for each property value the time of inclusion, the data source and the linking method used.

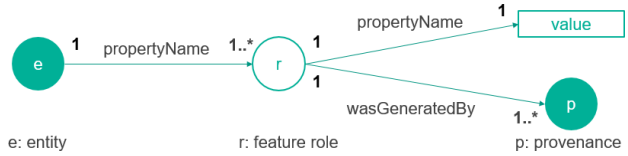

Fig. 3. Knowledge graph data model.

We construct the initial knowledge by creating a new entity for each entity in the data source, with a unique URI within the knowledge graph and a `schema:sameAs` property pointing to the original URI.

The knowledge graph is extended by adding only the properties and property values of the data source which do not exist yet for the relevant entity. Given the data of the existing knowledge graph D_{kg} and the data of the subsequent data source D_s, the extended knowledge graph D^e_{kg} holds the data from D_{kg}

[6] `http://www.w3.org/TR/prov-o/`

with data from D_s which was not there before $D^e_{kg} = D_{kg} \cup D_s$. Figure 4 shows how the consolidation of a subsequent data source would add an already existing name value (A), a new name value (B) and a new birth date value (C) to an entity in the knowledge graph. The addition of the existing alternative name (A) is represented by the added provenance to the existing feature role.

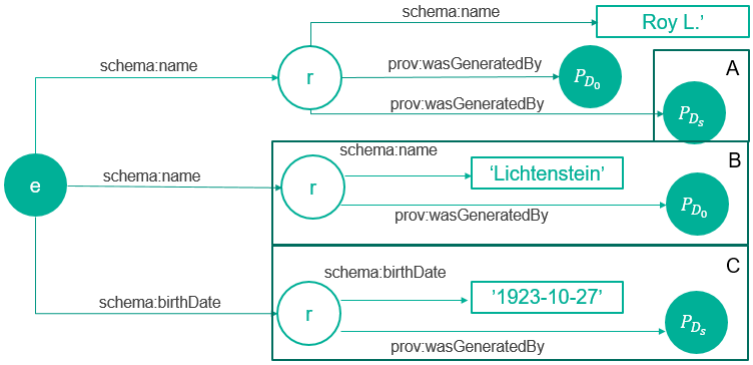

Fig. 4. Addition of a feature.

After all matched candidates are processed, the entities from the subsequent data source that were not matched to any existing entity in the knowledge graph need to be consolidated. Hence, we iterate over the data source and include each unmatched entity as a new entity in the knowledge graph.

4 Evaluation

We evaluate our approach by assessing the quality of the results and the run-time performance.

4.1 Ground truth

To quantify precision and recall, we manually build up a ground truth[7]. The ground truth holds links for the alphabetically first 200 artist entities which are represented in each of the four data sources. Because each entity exists in ULAN and SAAM, we know that the entities are artists even if they are not typed as an artist in DBpedia or VIAF. If a data source includes duplicates for an entity, these duplicates are included in the ground truth as well. Hence, links found by the approach that are not in the ground truth can be assumed to be wrong.

[7] `https://bitbucket.org/GlebGawriljuk/aifb-isi-knowledgegraphconstruction/raw/168b6ec21654e1de01d546567f7232b77daaf1a2/groundTruth_final_2015.tsv`

4.2 Quality evaluation

To evaluate the quality of the results, we review the steps of the approach individually for each source, starting with the candidate generation (LSH). We apply two matching functions in sequence. First, we apply the Hybrid-Jaccard function to check the similarity of the name values between the candidate and the target entity and only keep the candidates with a similarity above 0.5. Second, we apply the 'birth year' function to check whether the candidate has the same birth year as the target entity.

We start with the ULAN data and create a total of 109,414 entities in the initial knowledge graph. Consolidating the 8,406 SAAM entities, we link 3,443 entities and introduce 4,963 new entities, resulting in 114,377 knowledge graph entities. Then, we identify 58,944 DBpedia artist entities, link 34,838 of them, and introduce 24,106 as new entities, leading to 139,396 knowledge graph entities. Lastly, we identify 1,798,285 VIAF artist entities; we link 1,775,303 entities and introduce 22,982 as new. We merge many VIAF entities, as the VIAF data contains many duplicates and incomplete data (i.e., some records only have a last name) and our matching is solely based on the first and last name. The final knowledge graph holds 161,465 entities.

Table 3 shows the results of the ground truth evaluation based on the following quality measures:

- N: the number of ground truth entities which are introduced as new entities to the extended knowledge graph.
- T: the number of ground truth entities for which candidates are found.
- C: the number of candidates that are found for all ground truth entities.
- C_c: the number of correct candidates for all ground truth entities including duplicate candidates.
- C_t: the number of correct candidates for all ground truth entities excluding correct duplicate candidates.
- C_i: the number of incorrect candidates for all ground truth entities.

When evaluating precision and recall we have to consider that the knowledge graph can hold duplicate entities which are all correct candidates for a target entity. Therefore, we differentiate between the number of correct candidates including the duplicates (C_c) and the number of correct candidates excluding the duplicates (C_t). Both precision and recall measures are based on C_t. The precision can be calculated based on C_c as well, however, the precision would be always higher and would only differ in at most 1%.

In the candidate generation step we achieve a high recall and a low precision for all data sources. On the one hand, we miss only a few correct candidates due to a low LSH threshold, because the target's name differs from the candidate's name, due to a missing middle name, spelling errors or missing values. Therefore, both have a Jaccard similarity near or below our LSH threshold. Such candidates have only a chance of 50% or less to be found by the MinHash/LSH. On the other hand, the MinHash/LSH also finds incorrect candidates due to similar name value between the candidate and the target entity.

Table 3. Results of the ground truth quality evaluation.

			T	C	C_c	C_t	C_i	N	Recall	Precision
ULAN	Initial KG		-	-	-	-	-	202	-	-
SAAM	Candidate generation		200	1,841	201	200	1,640	0	100%	10.87%
	Linking	Hybrid-Jaccard	200	217	201	200	16	0	100%	92.59%
		birth year	187	188	188	187	0	13	93.50%	100%
DBpedia	Candidate generation		199	1,550	211	197	1,339	1	98.50%	12.83%
	Linking	Hybrid-Jaccard	197	226	208	196	18	3	98.00%	91.59%
		birth year	187	192	190	187	0	13	93.50%	100%
VIAF	Candidate generation		271	1,468	293	272	1,175	7	97.84%	18.80%
	Linking	Hybrid-Jaccard	271	369	290	270	79	0	97.12%	77.36%
		birth year	262	276	266	260	10	9	93.53%	96.30%

We anticipate to reduce the number of candidates and increase the precision by applying the Hybrid-Jaccard function on the found candidates. We manage to increase the precision close to 90% for all data sources, with a slight decrease in recall. This is achieved by removing incorrect candidates while keeping most of the correct candidates in the cluster. E.g., in the case of the SAAM consolidation, we decrease the number of candidates from 1,841 to 221 while keeping the number of correct candidates at 201.

We try to further increase the precision by applying the 'birth year'-function. Furthermore, we tolerate a decrease of the recall as a trade-off to the increased precision. We achieve to increase the precision for all data source consolidations. As foreseen, for all consolidation the recall decreases. The recall decreases because the 'birth year'-function removes correct candidates due to errors in the birth year values of the entities. An example is a correct candidate "Pietro Aquila" with "1592" as birth year value from ULAN[8] but "1650" from SAAM[9].

We do not achieve a 100% precision because some incorrect candidates always remain, in case an incorrect candidate has a similar name and no birth year value. Due to a similar name both the LSH and the Hybrid-Jaccard keep the candidate in the cluster. The 'birth year'-function cannot evaluate the candidate because of the missing birth year and, thus, the incorrect candidate stays in the cluster.

The results show that we manage to achieve a high recall during the candidate generation as anticipated. Furthermore, we increase the precision by applying the Hybrid-Jaccard-function. Lastly, we succeed in further increasing the precision for the cost of a lower recall by executing the 'birth year'-function.

We tolerate a decreasing recall for an increasing precision because we argue that a higher precision is more important than a high recall. With a high recall, we can be sure to find all correct candidates for the cost of creating noise in form of `schema:sameAs` links between non-matching entities. In such a case, each data

[8] `http://vocab.getty.edu/ulan/500018769`
[9] `http://edan.si.edu/saam/id/person-institution/121`

source added would lead to more noise, wrongly consolidating more entities into one entity in the knowledge graph. In contrast, by prioritizing a high precision we ensure that different entities are not consolidated into one entity, leading to less noise (but more potential duplicates) in the knowledge graph.

4.3 Performance evaluation

For performance evaluation we measure the run-time for each step and review the run-time in relation to the data size. Experiments are conducted on an Ubuntu machine with 4 AMD Opteron 62xx class 2GHz CPU cores and 32 GB RAM.

Table 4. Overall performance per data source in hours:min:sec.

Step	ULAN	SAAM	DBpedia	VIAF
Candidate generation	-	00:15:59	01:55:14	29:58:26
Linking	-	00:01:37	01:11:22	55:02:13
Consolidation	00:02:12	00:04:49	00:23:20	156:34:12
Total	00:02:12	00:22:25	03:29:56	229:00:39

Table 4 shows the run-time of each step of the approach for each data source. We find a long run-time for the Hybrid-Jaccard rule and the consolidation on the VIAF data source for two reasons. Firstly, the VIAF data source is significantly larger than the other data sources. Secondly, VIAF includes significantly more different name values for each entity in multiple languages and variations. Hence, the Hybrid-Jaccard rule must evaluate more values for each target entity when linking the VIAF entities. Overall we can observe an increasing run-time with each run as the size of both the knowledge graph and the data sources increases.

5 Related work

The challenge of automatic knowledge graph construction is a trending topic in current research[10]. The body of work can be divided into three groups.

The first group, such as DIG [16], Knowledge Vault [5], CiteSeerX [1], Pujara et al. [12], the Knowledge Graph Identification [13] and NELL [3], builds knowledge graphs with a fixed schema from data retrieved using information extraction techniques. Similarly, we build up the knowledge graph with a fixed target schema. However, we do not apply information extraction techniques to retrieve data from the web. Our focus is on using existing structured web data sources, consolidating their data, and providing an integrated view on the data from multiple data sources within a single knowledge graph.

The second group, such as Reverb [6], OLLIE [11], and PRISMATIC [7], uses open information extraction without a fixed schema. Again, our approach differs

[10] For example, the series of workshops on Automated Knowledge Base Construction (AKBC), `http://www.akbc.ws/`.

in the objective: we integrate structured data rather than extract data. Also, we operate on a fixed schema, while the open information extraction approaches assume no fixed schema.

The third group consolidates data from structured data sources to build up the knowledge graph. We consider our work strongly related to LDIF [14], YAGO [15], YAGO2 [9], and Freebase [2], which we now discuss in detail.

The Linked Data Integration Framework (LDIF) [14] designs a similar approach to construct knowledge graphs. Both LDIF and our approach consolidate web data into a knowledge graph by mapping input data to a target schema, linking entities, and consolidating the data based on the found links. The most significant difference is that LDIF uses a Hadoop implementation to parallelize the work of their transformation modules. In contrast, we scale by applying the MinHash/LSH algorithm. Furthermore, both approaches include provenance information in the knowledge graph. However, where LDIF includes only the origin of data, we additionally include the processing information.

YAGO [15] builds up a knowledge graph by defining entity classes from the conceptual Wikipedia categories and WordNet terms that do not represent an entity. However, entity data is included only from Wikipedia as all WordNet terms representing an entity are removed. Hence, no entity resolution among the entities in Wikipedia and WordNet is applied. YAGO2 [9] does apply entity resolution, but only on the location entities from Wikipedia and GeoNames[11] Similar to our approach, the entity resolution techniques can be seen as multiple rule-based matching functions. Their approach first matches the location entities based on their names. If multiple candidates are found, YAGO2 uses the data from the geo-coordinates of the entities to decide which candidate is a match. However, these rules are fixed part of the approach and cannot be exchanged.

Freebase [2] is a tuple database which is collaboratively created, structured, and maintained. Freebase provides an HTTP-/JSON-based API to maintain the data through read and write operations. However, entity linking is carried out in an interactive way with user input required for consolidation.

6 Conclusion

We have addressed the problem of efficiently building a consolidated knowledge graph of a specific entity type from data spread over multiple large data sources. Our approach works by aligning the data to a domain ontology, identifying the candidate links using the MinHash/LSH techniques, generating links with high precision using entity linking rules, and finally building the knowledge graph.

We have applied our approach to building a knowledge graph about artists containing 161,465 artists consolidated from four data sources. We have evaluated the resulting knowledge graph and showed that on a sample of data it has an average recall of 96,82% and a precision of 99,17%. We also have demonstrated that we are able to efficiently construct a knowledge graph of high quality, processing 17,539,125 entities in about 10 days on modest hardware.

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

References

1. G. Alexander, I. Ororbia, J. Wu, and G. C. L. Citeseerx: Intelligent information extraction and knowledge creation from web-based data. In *Proceedings of the 4th Workshop on Automated Knowledge Base Construction at NIPS*, 2014.
2. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2008.
3. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. J. Hruschka, and T. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*, 2010.
4. A. Doan, H. A., and Z. Ives. *Principles of Data Integration*. Elsevier, 2012.
5. X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.
6. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in NLP and Computational Natural Language Learning (EMNLP)*, 2011.
7. J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur. Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the 1st International Workshop on Formalisms and Methodology for Learning by Reading*, 2010.
8. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999.
9. J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence Journal*, 194:2861, January 2013.
10. C. Knoblock, P. Szekely, J. Ambite, A. Goel, S. Gupta, K. Lerman, P. Mallick, M. Muslea, and M. Taheriyan. Semi-automatically mapping structured sources into the semantic web. In *Proceedings of the 9th Extended Semantic Web Conference (ESWC)*, 2012.
11. M. Mausam, Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *Proceedings of the Conference on Empirical Methods on NLP and Computational Natural Language Learning (EMNLP)*, 2012.
12. J. Pujara and L. Getoor. Building dynamic knowledge graphs. In *Proceedings of the Knowledge Extraction Workshop at NAACL-HLT*, 2014.
13. J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. In *Proceedings of 12th International Semantic Web Conference*, 2013.
14. A. Schultz, A. Matteini, R. Isele, P. Mendes, C. Bizer, and C. Becker. Ldif - a framework for large-scale linked data integration graphs. In *Proceedings of 21st International Conference on World Wide Web*, 2012.
15. F. Suchanek, G. Kasneci, and G. Weikum. Yago - a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, 2007.
16. P. Szekely, C. A. Knoblock, J. Slepicka, A. Philpot, A. Singh, C. Yin, D. Kapoor, P. Natarajan, D. Marcu, K. Knight, D. Stallard, S. Karunamoorthy, R. Bojanapalli, S. Minton, B. Amanatullah, T. Hughes, M. Tamayo, F. Flynt, R. Artiss, S. Chang, T. Chen, G. Hiebel, and L. Ferreira. Building and using a knowledge graph to combat human trafficking. In *Proceedings of the 14th International Semantic Web Conference*, 2015.