DBkWik: extracting and integrating knowledge from thousands of Wikis

Sven Hertling1 · Heiko Paulheim1

Received: 2 January 2019 / Revised: 3 October 2019 / Accepted: 5 October 2019 / Published online: 2 November 2019
© Springer-Verlag London Ltd., part of Springer Nature 2019

Abstract
Popular cross-domain knowledge graphs, such as DBpedia and YAGO, are built from Wikipedia, and therefore similar in coverage. In contrast, Wikifarms like Fandom contain Wikis for specific topics, which are often complementary to the information contained in Wikipedia, and thus DBpedia and YAGO. Extracting these Wikis with the DBpedia extraction framework is possible, but results in many isolated knowledge graphs. In this paper, we show how to create one consolidated knowledge graph, called DBkWik, from thousands of Wikis. We perform entity resolution and schema matching, and show that the resulting large-scale knowledge graph is complementary to DBpedia. Furthermore, we discuss the potential use of DBkWik as a benchmark for knowledge graph matching.

Keywords Knowledge graph creation · Information extraction · Linked open data · Knowledge graph matching

1 Introduction

General purpose knowledge graphs, such as DBpedia, YAGO, and Wikidata, have become a central part of the linked open data cloud [49] and are among the most frequently used datasets within the Web of data [8]. Such knowledge graphs contain information on millions of entities from multiple topical domains [37].

Many of the popular knowledge graphs are created from Wikipedia and hence have a similar coverage [47]. Generally speaking, each real-world entity for which a dedicated Wikipedia page exists becomes an entity in the knowledge graph. This is a fundamental restriction for many applications—for example, for building content-based recommender systems backed by knowledge graphs, Di Noia et al. showed that the coverage of entities in popular recommender system datasets in DBpedia is no more than 85% for movies, 63% for music artists, and 31% for books [35].
In this paper, we introduce the DBkWik\(^1\) knowledge graph\(^2\) (licensed under the Creative Commons Attribution-Share Alike License\(^3\)). It is generated by applying the DBpedia extraction framework\(^4\)—i.e., the software that generates the DBpedia knowledge graph out of a Wikipedia dump—to thousands of Wikis from a Wikifarm. The result is a large-scale knowledge graph with more than 11M instances and more than 90M RDF triples, i.e., it contains twice as many instances as the Wikipedia-based knowledge graphs DBpedia and YAGO. Besides the dataset itself, we also release a set of gold standards for schema and instance matching, which have been used to tune and evaluate the methods for interlinking and knowledge fusion within DBkWik.

This paper is an extended version of a paper presented at the IEEE International Conference on Big Knowledge (ICBK) 2018\(^{[21]}\). The contributions of this paper are the following:

- The publicly available DBkWik dataset with more than eleven million entities
- A workflow for generating an integrated, large-scale knowledge graph from different semi-structured sources
- An in-depth analysis of that resulting dataset
- The new knowledge graph track at OAEI together with a gold standard, and
- A discussion of insights from the first iteration of that OAEI track

The rest of this paper is structured as follows. Section 2 describes the approach for creating DBkWik. Section 3 gives insights into the topology and contents of the resulting knowledge graph. Section 4 discusses the use of DBkWik as a benchmark for matching in a new knowledge graph at the ontology alignment evaluation initiative (OAEI)\(^{[1]}\). Afterward, we present related work in Sect. 5 and close with a discussion and an outlook on future developments.

## 2 Approach

Both DBpedia and DBkWik are created from dumps of MediaWiki,\(^5\) an open-source software for creating Wikis, which is, among others, used by Wikipedia. Hence, for creating the DBkWik knowledge graph, we use the same software which has been developed for generating DBpedia, i.e., the DBpedia Extraction Framework. From a high-level point of view, the DBpedia Extraction Framework takes a Wiki dump\(^6\) as input and produces a knowledge graph as output.\(^7\) In that knowledge graph, one instance is created for each Wiki page, and one triple is created for each entry in an infobox (e.g., the population or the capital of a country). Links in an infobox create a relation between two resources (e.g., a city and a country for a capital infobox link), whereas non-linked values in an infobox (e.g., numbers or dates) create a literal assertion (e.g., the population of a country).

Besides the Wikipedia dumps, the DBpedia extraction framework also takes as input an ontology and a set of mappings between Wiki elements (i.e., infoboxes and keys used

---

\(^1\) Pronounced dee-bee-quick.

\(^2\) http://dbkwik.org/.

\(^3\) http://creativecommons.org/licenses/by-sa/3.0/.

\(^4\) https://github.com/dbpedia/extraction-framework.

\(^5\) https://www.mediawiki.org/wiki/MediaWiki.

\(^6\) https://meta.wikimedia.org/wiki/Data_dumps/Dump_format.

\(^7\) In the scope of this work, we restrict ourselves to Wikis created using the MediaWiki software, but this is merely a technical, not a conceptual limitation—as long as a Wiki software is able to create reasonably structured Wikis, e.g., allows to create infoboxes, categories, etc., it could be used as a source for knowledge graph creation.
DBkWik: extracting and integrating knowledge from thousands of Wikis...

Fig. 1 The overall workflow creating the DBkWik knowledge graph

within those infoboxes) and that ontology. Those mappings are used to create a more strictly formalized subset of Wikipedia. Moreover, it is used to assign types to instances. For DBpedia, the ontology and the mappings are created manually in a collaborative workflow.

Since neither manually created mappings nor a central ontology exists for DBkWik—and it would be infeasible to manually map thousands of Wikis to such an ontology—we have to take a different approach here. We extract a very shallow schema for each Wiki, creating a property for each infobox key and a class for each type of infobox. Later in the process, we identify identical classes and properties using schema matching, and we statistically infer subclass relations as well as domain and range restrictions.

Furthermore, in a knowledge graph based on a single Wiki, there is usually no more than one Wiki page for each real-world entity. Hence, when creating a resource for each Wiki page, duplicate resources will not occur. When applying the approach to a multitude of Wikis, different Wikis may overlap and thus have a page describing the same real-world entity. Therefore, the resulting extraction will not be free of duplicates; hence, duplicate detection (i.e., matching the knowledge graphs extracted from the individual Wikis to each other) and data fusion (i.e., unifying the information about entities in different Wikis) must be performed in addition to the extraction.

In addition to the pure extraction and fusion, we also apply two knowledge graph refinement operations [37], i.e., type induction using a simplified version of SDType [40], and statistical schema enrichment including subclass relations as well as domain and range restrictions [52]. The final dataset is loaded into a Virtuoso server [9], which makes the knowledge graph available both as a Linked Data service as well as a SPARQL endpoint. Figure 1 shows the overall workflow of the DBkWik generation.

2.1 Extraction of the initial knowledge graph

The initial knowledge graph is extracted from a set of Wiki dumps using the DBpedia extraction framework. We created our own, modified version of the DBpedia extraction framework.

8 https://github.com/sven-h/extraction-framework.
to cope with two issues, i.e., (a) the aforementioned missing mappings and central ontology, and (b) the fact that we want to process dumps from arbitrary MediaWiki versions and configurations, while the original extraction framework is tailored toward the specific configuration of MediaWiki which is used by Wikipedia. Specifically, the following changes were made:

- The hard coded URI prefix http://dbpedia.org was replaced by individual URIs per Wiki
- All mapping-based extractors were removed (cf. a)
- Types and properties in the schema are automatically created from infobox template names and infobox keys instead (cf. a)
- Abstracts are extracted using the Sweble parser [6] rather than by setting up MediaWiki instances (cf. b)

The result is one knowledge graph per input Wiki. While links to other Wikis may exist for a small set of pages in different Wikis (and they are extracted where available), the result is a set of mostly disconnected individual knowledge graphs (one per input Wiki), with a small set of interconnections.

2.2 Linking to DBpedia

To make DBkWik a five star dataset [17] and a proper citizen of the linked open data cloud [49], we include interlinks to DBpedia, which serves as a central interlinking hub for many datasets.

We include interlinks both on the schema as well as on the instance level. For interlinking on the schema level, we currently use case-insensitive string equality and break ties by the closest case sensitive match. While more sophisticated interlinking approaches are possible, a preliminary study on previous versions of the dataset has shown that this approach already guarantees a high quality mapping with an $F_1$ score above 0.85 [23].

In the same preliminary study, we have observed that string-based matching alone does not guarantee a suitable mapping on the instance level. Therefore, for interlinking on the instance level, we pursue a different approach.

We leverage the fact that both DBkWik and DBpedia are generated from Wikis; hence, we can also use the Wiki page itself for the mapping. While a Wiki page may or may not contain an infobox (i.e., relations for the corresponding entity may or may not be extracted), there is usually a reasonable amount of text on the page. Therefore, we assume that the Wiki page text gives the better signal.

To perform the matching, we use the short and long abstracts extracted by the DBpedia Extraction Framework: the short abstract is the first paragraph of a Wiki page, the long abstract is all text which occurs before the first intermediate headline [19]. We train a doc2vec model [29] on all abstracts extracted from Wikipedia and the Wiki collection used as input for DBkWik. That model assigns a point in a high dimensional vector space to each abstract (and hence to each entity in DBkWik and DBpedia). We experimented with both approaches proposed in [29], i.e., PV-DM and PV-DBOW, and both long and short abstracts. For finding matching candidates for DBkWik entities, we first find DBpedia candidates derived from a page with the same title (or a redirect from the same title), and then pick the candidate with the largest cosine similarity in the corresponding doc2vec space.
2.3 Internal linking and fusion

The same approaches (i.e., string-based matching on the schema level, exploiting doc2vec on the instance level) are used to link and fuse the dataset internally, i.e., to identify duplicate instances, classes, and properties. As a result, we can unify the schema, i.e., create a common set of classes and properties, as well as fuse all entities which are identified to be equal into one. To that end, new URIs are created using a hash function on the concatenated original URIs.

Formally, DBkWik is created as a quotient graph [15] from the original extraction, given an equivalence relation for both the schema and instance level, which is created using matching heuristics in our setting. In this new graph, all instances and schema elements from the same equivalence class are replaced by a new, fresh URI.

As a result, we obtain a knowledge graph where all information about duplicate entities is fused into one entity. The resulting knowledge graph has less entities, but is more strongly connected. Figure 2 illustrates that process.

2.4 Schema and type enrichment

After the fusion, we perform two more steps. The first step aims at further enriching the so far very shallow ontology. Here, we follow the approach introduced in [52], using association rule mining for determining both subclass relations as well as domain and range restrictions. Following the observation that in a skewed dataset like a general purpose knowledge graph,
a minimum support threshold is not very meaningful [38], we impose no minimum support threshold and a minimum confidence of 0.95.

In the example in Fig. 2, given that we observe many instances like dbkwik:a7f9bc02, which have both types dbkwik:Artist and dbkwik:Person, we would induce the subclass relation

\[
\text{dbkwik:Artist rdfs:subclassOf dbkwik:Person}
\]

More formally: given that we impose a minimum confidence threshold of 0.95, we add the subclass relation to the schema if 95% of all instances of dbkwik:Artist are also instances of dbkwik:Person.

The subclass relations are used directly for materialization. Therefore, in the above example, we would add the triple

\[
\text{dbkwik:87a0c82f rdf:type dbkwik:Person}.
\]

With the same mechanism, domain and range restrictions are added to the schema: following the example in Fig. 2, and considering the property dbkwik: birthPlace, we add dbkwik:Person as the property’s domain if 95% of all instances in the subject position have dbkwik:Person as their type, and dbkwik: Place as the property’s range if 95% of all instances have dbkwik:Place as their type.

Those domain and range restrictions are used to implement a light-weight version SDType [40], using the distribution of inferred domain and range restrictions instead of the actual distributions to allow for a more efficient implementation. While the original implementation of SDType uses the actual distribution of relations and types to assign probabilities for types, i.e.,

\[
\text{SDType}(o, c) = v \cdot \sum_{(s, p, o) \in G} w_p \cdot P((o, \text{type}, c) \in G \mid (x, p, o) \in G),
\]

using a normalization factor \(v\) and property weights \(w_p\) which are depending on the specificity of the properties, we use the inferred ranges instead for a simplified and more efficient computation:

\[
\text{SDLight}(o) = \arg \max_{c \in \text{classes}(G)} |\{(p|(s, p, o) \in G \land (p, \text{range}, c) \in G)\}|
\]

where \(o\) is the instance for type prediction, \(c\) the class, and \(P\) the conditional probability. \(G\) represents the knowledge graph consisting of a set of triples.

### 3 Analysis of the resulting dataset

Following the methodology described above, we applied the approach to Fandom powered by Wikia\(^9\), which is one of the most popular Wiki Farms\(^10\), comprising more than 423,000 individual Wikis totaling nearly 40 million articles.

\(^9\) http://fandom.wikia.com/.

\(^10\) http://www.alexa.com/topsites/category/Computers/Software/Groupware/Wiki/Wiki_Farms.
3.1 Input data

Only a small fraction of Wikis at Fandom has a dump that is downloadable. Dumps are not created automatically, but only upon request by the Wiki owner. In total, we were able to obtain a total of 12,840 dumps comprising 14,743,443 articles.

Fandom publishes Wikis in various languages, and each Wiki comes with two orthogonal topical classifications, called topics and hubs, where the former are more fine grained than the latter. Figure 3 shows a breakdown by language, topic, and hub of the Wikis for which we could download a dump. It can be observed that the majority of the Wikis is in English, while topic-wise, games, and entertainment-related Wikis are predominant.

Figure 4 shows the articles per Wiki in a log log plot. The empirical data is best fitted with a heavy-tailed distribution like a power law or lognormal distribution. This means in Fandom are only a few Wikis with a lot of articles but also a lot of Wikis with only a few articles. With the help of the powerlaw package [2] which is based on [5], we have tested the following distributions: exponential, lognormal, lognormal (positive), power law, stretched exponential, and truncated power law. The best fitting one is lognormal with $\mu = -10.617$ and $\sigma = 4.446$ followed by the power law with $\alpha = 2.030$. The starting point $x_{\text{min}}$ is optimized for both cases to be 2648.

Table 1 shows the top ten English Wikis sorted by their dump size. Some Wikis have a large amount of text (corresponding to the dump size) but only sparse structured information.
Fig. 4 The complementary cumulative distribution function of the amount of articles per Wiki. Two heavy-tailed distributions are fitted to the empirical data.

Table 1 Top ten English Wikis sorted by dump size together with the amount of extracted triples

| Wiki URL     | Hub       | Topic     | Dump size (MB) | #Triples    |
|--------------|-----------|-----------|----------------|-------------|
| speedydeletion | Lifestyle | Lifestyle | 4226           | 5,587,303   |
| military      | Lifestyle | Education | 2635           | 52,635,203  |
| allthetropes  | TV        | Movies    | 1583           | 29,134,472  |
| answers       | Lifestyle | Education | 1117           | 2,902,491   |
| scratchpad     | Lifestyle | Creative  | 908            | 11,632,570  |
| marvel        | Comics    | Comics    | 800            | 2,198,267   |
| starwars      | Movies    | Entertainment | 572       | 38,279,474  |
| heykidscomics | Comics    | Entertainment | 560       | 18,553,366  |
| eq2           | Games     | Gaming    | 438            | 6,142,610   |
| dc            | Comics    | Lifestyle | 421            | 4,904,611   |

in infoboxes (which corresponds to the amount of extracted triples in each Wiki). Some examples are speedydeletion, answers, marvel, eq2, and dc. For Wikis like heykidscomics and starwars, the opposite holds. The dump size is about 500 MB, but more than 18 million triples can be extracted.

3.2 The DBkWik knowledge graph

The initial creation of the knowledge graph (i.e., running the DBpedia extraction framework on the set of downloaded dumps) leads to an initial, unreconciled knowledge graph. Table 2 depicts the characteristics of that initial knowledge graph extracted (column initial).

The internal matching and fusion reduces the number of instances by a factor of 21.4%. On the schema level, the reduction is even larger, with the number of classes reduced by 83.2%, and the number of properties reduced by 74.6%. In total, the number of statements is reduced by a factor of 15.1%. Note that this is the smallest factor, because a statement $s \ p \ o$ is removed if and only if there is a statement $s' \ p' \ o'$, where all three of the following apply: $s$ is mapped to $s'$, $p$ is mapped to $p'$, and $o$ is mapped to $o'$. At the same
Table 2  Characteristics of the initial and final knowledge graph

|                  | Initial            | Final               |
|------------------|--------------------|---------------------|
| # Instances      | 14,212,535         | 11,163,719          |
| # Typed instances| 1,880,189          | 1,372,971           |
| # RDF Triples    | 107,833,322        | 91,526,001          |
| Avg. indegree    | 0.624              | 0.703               |
| Avg. outdegree   | 7.506              | 8.169               |
| # Classes        | 71,580             | 12,029              |
| # Properties     | 506,487            | 128,566             |

Table 3  Most frequently matched classes, properties, and instances

| Classes   | Name      | #match | Properties  | Name      | #match | Instances  | Name        | #match |
|-----------|-----------|--------|-------------|-----------|--------|------------|-------------|--------|
|           | Character | 5036   | Name        | 4131      | Main_Page | 2528       |             |        |
|           | Episode   | 4261   | Title       | 3013      | Current_events | 1188      |             |        |
|           | Location  | 4106   | Caption     | 2303      | Templates  | 1096       |             |        |
|           | Album     | 4075   | Type        | 1981      | Characters | 642        |             |        |
|           | Item      | 4069   | Gender      | 1962      | Timeline   | 489        |             |        |
|           | Quest     | 3869   | Status      | 1611      | Earth      | 451        |             |        |
|           | Event     | 3750   | Next        | 1566      | Human      | 418        |             |        |
|           | Book      | 1365   | Location    | 1488      | Dragon     | 377        |             |        |
|           | Box       | 852    | Header      | 1475      | Weapons    | 376        |             |        |
|           | Film      | 402    | Body        | 1463      | Hauptseite | 373        |             |        |

time, the density of the graph increases, with the average indegree and outdegree of resources increasing by 12.6% and 8.8%, respectively.

Table 3 shows the 10 classes, properties, and instances which are most frequently matched. For example, the class character is extracted from more than five thousand Wikis, and the property name is extracted from more than four thousand, and all of those are mapped to a common property. Among the top 10 instances, there are pages which are likely to occur in many Wikis (such as Main Page or Templates), but also interesting instances such as Earth (extracted from 451 Wikis), Human (extracted from 418 Wikis), and Dragon (extracted from 377 Wikis). Consequently, the fused instances have very large degrees in the final knowledge graph (Earth: 13,518, Human: 38,311, Dragon: 3391).

Figure 5 shows the outdegree for the largest classes in DBkWik. For the first four classes (character, item, episode, and song), the number of statements for an instance is very similar, whereas for location it is a bit lower. For the classes actor and character, the variance is very high. This implies that some instances have a lot outgoing edges and some only a few. The reason is probably that some instances of actor or character appear in multiple Wikis and thus have more information, whereas long tail entities (not well known entities) might appear only in only one or a few Wikis, and thus, the merged concept only has some information.

It can also be observed that in the current version of DBkWik, cross-language fusion of entities does not always work, as observed by the two entries Main_Page and Hauptseite.\footnote{Hauptseite is German for main page.}
Applying the schema induction approach based on association rule mining, we obtain a total of 5,347 class subsumption rules, 58,724 domain restrictions and 114,272 range restrictions (i.e., 45.7% of all properties are assigned a domain, and 88.9% are assigned a range). In total, materializing the subclass relations leads to 626 additional `rdf:type` statements. After applying the light-weight type induction based on SDType, we obtain another 97,293 `rdf:type` statements to the knowledge graph.

### 3.3 Linkage quality

To evaluate the quality of the interlinking to DBpedia, as well as the internal linkage quality, we have created two gold standards. For the evaluation of the interlinking to DBpedia, an already created gold standard [23] is used. It contains links to DBpedia for eight Wikis both on the instance and the schema level. It was originally generated by three domain experts and partially improved because some inconsistencies were noticed. More specifically, it shows matches between a random resource in one Wiki and (if available) the corresponding resource in DBpedia. This is done for classes, properties as well as instances. The information that a resource is not matched to DBpedia is also stored and used for the evaluation. In case a matching system returns a correspondence between two resources where in the gold standard at least one of it should not be matched, the correspondence is counted as a false positive. Since the assessment of false positives is not trivial in the presence of a gold standard which is only partial [44], this inclusion of explicit negative mappings in the gold standard helps fostering a more accurate assessment.

The resulting interlinking quality is shown in Table 4. It shows the precision, recall, and F1-measure for classes, properties, and instances. For instance matching, five different approaches were evaluated, with string similarity used as a baseline. For the four approaches using doc2vec, we also depict the optimal threshold. DM stands for the distributed memory model and DBOW for distributed bag of words. These are the main two variants of doc2vec.
The schema-level links were created manually by ontology experts. For the instance level links, we set up a crowdsourcing task using Amazon MTurk. To ensure that a significant number of links can be identified, we picked three English-language Wikis each from three topics (see Table 5) which shared a high overlap of labels, and sampled 30 pages from each Wiki. Workers were asked to identify matching pages in the other two Wikis from the same topic. Each individual task (i.e., finding matching pages in two Wikis) was performed by five crowd workers. The inter-rater agreement according to Fleiss’ kappa [13] is 0.8762, which is an almost perfect agreement according to [28].

13 The short abstract is generated by extracting the first text paragraph from a Wiki page, whereas the long abstract is all text before the first headline.

14 We restricted the workers to have a 95% approval rate and a minimum of 100 approved HITs (human intelligence tasks), following the recommendations by [25] and [16], and restricted their location to the USA to attract a large fraction of native speakers. We paid $0.40 for a HIT of finding matching pages for 10 pages in two Wikis. In total, the creation of the gold standard took 10 days. Details on the task design as well as the resulting gold standard are available online at https://github.com/sven-li/dbkwik.
The results of the evaluation of the internal interlinking are shown in Table 6, using the thresholds for doc2vec that worked best for the linking to DBpedia. It can be observed that the schema-level matching is in a similar range as for the linking to DBpedia, while the instance-based interlinking using doc2vec does not significantly improve over string-based matching.

At first glance, the latter is a surprising observation. However, we assume that the main reason is a bias in our gold standard, since the Wikis are chosen in a way that matches are likely to occur, i.e., they share similar topics. This means that in this gold standard, it is not likely that two pages with the same title describe two different instances, while this is likely in the general case of matching to arbitrary Wikis. Therefore, we expect that, although string similarity works fine on the gold standard, it is likely to produce more false positives in the general case. Hence, we stuck to the doc2vec based approach also for the internal linking and fusion for creating the final dataset.

### 3.4 Complementarity to DBpedia

In [47], we have introduced a method for estimating the overlap of two knowledge graphs, given that (a) an imperfect link set between the two exists, and (b) we can estimate the quality of that link set in terms of precision and recall. We propose that given that there exist \( N \) links at a precision of \( P \) and a recall of \( R \), the two knowledge graphs have an overlap \( O \) (i.e., number of common entities) of

\[
O = N \cdot P \cdot \frac{1}{R}
\]

Since the best mapping approach to DBpedia found 552,292 links at a precision of 0.643 and a recall of 0.672, the total estimated overlap is 528,458. In other words, only 4.7% of all entities in DBkWik are also contained in DBpedia. Likewise, 10.3% of all entities in DBpedia are also contained in DBkWik. These numbers illustrate the high complementarity between DBpedia and DBkWik.

In addition, we also used the schema-level class mappings to DBpedia to identify classes that are considerably larger in DBkWik than in DBpedia. Table 7 depicts the largest classes in DBkWik and the corresponding instance counts in DBkWik and DBpedia. It can be observed that while for three out of the ten classes (the gaming specific classes `item`, `quest`, and `jutsu`), there is no corresponding class in DBpedia, there are at least four additional classes (fictional...
Table 7 The largest classes in DBkWik with instance counts in DBkWik and DBpedia

| Class                  | # Instances DBkWik | # Instances DBpedia |
|------------------------|--------------------|---------------------|
| Fictional character    | 303,598            | 21,845              |
| Item                   | 82,395             | –                   |
| Episode                | 56,048             | 10,633              |
| Location               | 24,582             | 881,597             |
| Song                   | 22,436             | 9225                |
| Actor                  | 20,278             | 6697                |
| Ship                   | 18,591             | 35,486              |
| Jutsu                  | 18,122             | –                   |
| Military Person        | 17,577             | 33,181              |
| Quest                  | 17,272             | –                   |

– indicates that no corresponding class exists in DBpedia

character, episode, song, and actor) where DBkWik has significantly more instances than DBpedia. On the other hand, locations, ships, and military persons are better covered by DBpedia.

4 OAEI knowledge graph track

Fusing the schema and instances of the individual knowledge graphs extracted from the different Wikis is an essential step in making DBkWik a unified knowledge graph. However, as the results above show, the matching required for the fusion is a non-trivial task. It poses interesting challenges, both in terms of result quality as well as in terms of scalability. Therefore, we decided to use the crowdsourced gold standard for creating a new track in the ontology alignment evaluation initiative (OAEI).

The purpose of OAEI is the systematic benchmarking and evaluation of ontology matching approaches and tools. Every year, new systems or problems can be submitted. Usually, the tracks focus on either T-Box/Schema matching or instance matching. For OAEI 2018, the generated gold standard for Wiki interlinking in this paper was used to form new track. In the first place, our aim is to analyze how existing matching tools can handle the task at hand. In the long run, we want to encourage the development of matching tools specifically developed for the task of knowledge graph matching.

The main difference between the existing tracks of OAEI and the new knowledge graph track is the simultaneous matching of instances, classes, and properties. It is interesting to see if already available systems with potential modifications can deal with such a problem and how well they are doing.

4.1 Data preparation and evaluation routines

We had to modify the knowledge graphs as well as the evaluation routine for participating at OAEI. When running a task, a matching system is provided with two knowledge graphs

15 In many cases, this can be explained by the notability criteria of Wikipedia, cf. http://en.wikipedia.org/wiki/Wikipedia:Notability.
formatted as RDF/XML. We had to modify the property URIs such that the transformation to RDF/XML was possible. Furthermore, the following information is condensed to one file: all categories for an article, the category labels and hierarchy, disambiguations, external links, images, all triples extracted from the infobox and their definitions, short and long abstracts, page titles and the type of resource extracted from the infobox as well as their definitions. Many of the systems require that every instance has a corresponding type. This is not the case when extracting the type information from infoboxes. Thus, an abstract type is created and assigned to every untyped instance.

The evaluation system for OAEI is called SEALS [10]. It is used for running the matching systems and to produce the resulting alignment files. Based on the fact that the created gold standard is only partial, the evaluation function has to be modified. In our evaluation, we follow the notion of recall and precision definitions for partial gold standards as defined in [44]: In case a mapping of the form \(<A, \text{null}, =, 1.0>\) is contained in the gold standard (which means resource A has no correspondence) then all mappings which have A as a source become a false positive. A further assumption is that in one Wiki only one article (instance) represents a concept. Thus, a mapping \(<A, A', =, 1.0>\) in the gold standard allows to penalize the mapping \(<A, B, =, 1.0>\) because in the second Wiki no similar resource of A' exists. This notion is similar to the local closed world or partial completeness assumption [7,14].

4.2 Results from OAEI 2018

In total, there were seven tools participating in OAEI which produced results for the knowledge graph track: AML [11], POMAP++ [27], Holontology [48], DOME [22], and three versions of LogMap [24]. In addition, string similarity is employed as a benchmark. AgreementMaker Light (AML) combines lexical similarity measures with schema information such as domain and range of properties. POMAP++ is specialized to biomedical ontologies and combines ontology partitioning and machine learning for an efficient matching. Holontology combines seven steps like preprocessing and lexical matching with Levenshtein, Jaccard, and Lin, which are then passed to a constraint solver to find an optimal global solution. DOME is the implementation of our matching approach introduced above, with the small modification that doc2vec is only used if string equality on labels and fragments does not already create a similarity. The main purpose of this modification is runtime improvement. LogMap combines lexical similarity with logical reasoning.

The results are shown in Table 8. For classes, many tools reach good results, with Holontology leading the field, while properties are not matched by any tool but DOME. The reason is twofold. First, since many tasks in OAEI are put a lot of emphasis on matching classes, some of the tools are optimized toward this task. Second, and most prominently, DBkWik does not distinguish between owl:DatatypeProperty and owl:ObjectProperty, but only type properties as rdf:Property. The reason for this is that we can only infer a very shallow schema, and moreover, many properties are used in a dual way, having both literals and resources as objects. Hence, we cannot explicitly type those properties as datatype or object properties without violating the semantics of OWL. Obviously, most matching systems participating do not support the matching of properties typed as rdf:Property.

Overall, we can observe that for class matching, techniques that have strong lexical matching components, such as LogMap and Holontology, are superior to those more focused on structural matching, since the input ontologies are rather shallow. Some findings, however, are rather surprising. LogMapBio and POMAP++, which both exploit biomedical ontologies, work well on this setup which does not have any biomedical data, at least not in the gold
Table 8  Results for the OAEI knowledge graph track in 2018 for each participating system divided into class, property, instance, and overall mappings [1]

| System          | Time  | Tracks | Size  | Prec.  | F-m.   | Rec.   |
|-----------------|-------|--------|-------|--------|--------|--------|
| **Class performance** |       |        |       |        |        |        |
| AML             | 88,448| 5      | 11.6  | 0.85   | 0.64   | 0.51   |
| POMAP++         | 438   | 9      | 15.1  | 0.79   | 0.74   | 0.69   |
| Holontology     | 318   | 9      | 16.8  | 0.80   | 0.83   | 0.87   |
| DOME            | 13,747| 9      | 16.0  | 0.73   | 0.73   | 0.73   |
| LogMap          | 14,083| 7      | 21.7  | 0.66   | 0.77   | 0.91   |
| LogMapBio       | 2340  | 9      | 22.1  | 0.68   | 0.81   | 1.00   |
| LogMapLt        | 500   | 6      | 22.0  | 0.61   | 0.72   | 0.87   |
| Baseline        | 412   | 9      | 18.9  | 0.75   | 0.79   | 0.84   |
| **Property performance** |       |        |       |        |        |        |
| AML             | 88,448| 5      | 0.0   | 0.00   | 0.00   | 0.00   |
| POMAP++         | 438   | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| Holontology     | 318   | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| DOME            | 13,747| 9      | 207.3 | 0.86   | 0.84   | 0.81   |
| LogMap          | 14,083| 7      | 0.0   | 0.00   | 0.00   | 0.00   |
| LogMapBio       | 2340  | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| LogMapLt        | 500   | 6      | 0.0   | 0.00   | 0.00   | 0.00   |
| Baseline        | 412   | 9      | 213.8 | 0.86   | 0.84   | 0.82   |
| **Instance performance** |       |        |       |        |        |        |
| AML             | 88,448| 5      | 82,380.9 | 0.16  | 0.23  | 0.38  |
| POMAP++         | 438   | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| Holontology     | 318   | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| DOME            | 13,747| 9      | 15,688.7 | 0.61  | 0.61  | 0.61  |
| LogMap          | 14,083| 7      | 97,081.4 | 0.08  | 0.14  | 0.81  |
| LogMapBio       | 2340  | 9      | 0.0   | 0.00   | 0.00   | 0.00   |
| LogMapLt        | 500   | 6      | 82388.3 | 0.39  | 0.52  | 0.76  |
| Baseline        | 412   | 9      | 17743.3 | 0.59  | 0.69  | 0.82  |
| **Overall performance** |       |        |       |        |        |        |
| AML             | 88,448| 5      | 10,2471.1  | 0.19  | 0.23  | 0.31  |
| POMAP++         | 438   | 9      | 16.9  | 0.79   | 0.14   | 0.08   |
| Holontology     | 318   | 9      | 18.8  | 0.80   | 0.17   | 0.10   |
| DOME            | 13,747| 9      | 15,912.0 | 0.68  | 0.68  | 0.67   |
| LogMap          | 14,083| 7      | 97,104.8 | 0.09  | 0.16   | 0.64   |
| LogMapBio       | 2340  | 9      | 24.1  | 0.68   | 0.19   | 0.11   |
| LogMapLt        | 500   | 6      | 88,893.1 | 0.42  | 0.49   | 0.60   |
| Baseline        | 412   | 9      | 17,976.0 | 0.65  | 0.73   | 0.82   |

standard. Moreover, LogMapBio works better than the other two variants of LogMap without the biomedical components. The reasons for this are unknown and subject to future studies.

On the instance level, only four matchers—AML, DOME, LogMap, and LogMapLt, can provide results. Here, DOME provides the best results, since it exploits an external signal (i.e.,
the Wiki articles from which the instances were created) rather than structural information. However, for properties and instances, no approach beats the string similarity baseline.

That last observation hints at a shortcoming of our current gold standard and its construction. We assume that many of the crowdworkers that contributed to the gold standard only searched for the corresponding entities by name. Hence, the gold standard contains only few examples of matched instances with completely different names. Moreover, the pre-selected knowledge graphs all have the same topic, which also leads to few cases of homonyms, i.e., non-matched entities with the same name, as those would are more likely to occur across topics. In fact, when analyzing the gold standard used for OAEI 2018, we found that more than 90% of all mappings in the gold standard are trivial string equivalences, which leads to an overly simple task which does not reflect the complexity of the underlying matching problem.

4.3 Refined gold standard for OAEI 2019

Therefore, for OAEI 2019, we explored new ways of generating the gold standard is used. The 2018 version was generated by a crowdsourcing approach where each worker gets a concept from one Wiki and has to search the same concept in another Wiki. This approach has some disadvantages: (1) the amount of correspondences scales with the amount of money the workers get (2) to have the same distribution of easy and difficult correspondences, a random sample of resources is not sufficient (3) getting many interesting correspondences (no simple string matches) requires a lot of samples (4) it is not known beforehand if a resource can be matched or not. All in all this means that a very high amount of resources would have to be sampled to get a reasonable gold standard because many resources will not be matched at all. The one which actually have a correspondent, are likely matched with resources with the same or similar label. Only a few percentage are matches which requires more knowledge.

Another idea was to provide the workers with candidate pairs, and let them judge if it is correct or not. This requires a good matching system with a high recall because the precision is increased by the workers. As a matching system, the known web search engines can be used. It is found out that Google may not index all pages from Fandom Wikis and some search engines returns better results than others. The search engine of DuckDuckGo has an api, but it does not return results in the same way as the browser version, while using the browser version of DuckDuckGo programmatically leads to blocking the requester in case of too many requests. Assuming that the search engines provide also some non-trivial matches, this approach could be used to generate the gold standard. One downside is that the judgement should be done by multiple workers, which increases the amount of work to be collected via crowdsourcing.

Currently, the best way to generate the gold standard is data driven and also used for OAEI 2019. Sometimes the Wikis have links to identical concepts in other Wikis. This information is explored and used to generate the gold standard. The section has often a title like External

---

16 Given request “site:darkscape.fandom.com” on Google (approximately 2660 results) and Bing (approximately 42,500 results)—tested on 8/8/2019.
17 Request: “bear site:darkscape.fandom.com” Google: (1) Penguin Hide and Seek/Spawn locations (2) Spria (3) Joseph and Bing: (1)Bear ribs (2) Grizzly bear cub (3) Grizzly bear.
18 https://duckduckgo.com.
19 https://duckduckgo.com/api.
Table 9: Number of correspondences in the OAEI 2019 knowledge graph track

| Wiki source                        | Wiki target  | String equivalents | Non-trivial matches | % non-trivial matches |
|------------------------------------|--------------|--------------------|---------------------|-----------------------|
| marvelcinematicuniverse            | marvel       | 941                | 726                 | 43.6                  |
| memory-alpha                       | memory-beta  | 7209               | 2154                | 23.0                  |
| memory-alpha                       | stexpanded   | 1496               | 283                 | 15.9                  |
| starwars                           | swg          | 591                | 530                 | 47.3                  |
| starwars                           | swtor        | 1200               | 229                 | 16.0                  |
| Total                              |              | 11,437             | 3922                | 25.5                  |

Links, and thus all links to other Wikis in a section with a title containing link are used. In case multiple links from one page to another Wiki are found, all of them are removed (because an automatic disambiguation is not possible here). Similarly, links are removed when multiple pages from one Wiki point to a same page in another Wiki. Thus, a 1:1 mapping is ensured. All of the resulting links form the gold standard used for OAEI 2019. Table 9 shows the new test cases together with the counts for the gold standard.

5 Related work

In general, knowledge graphs can be created by various means, including manual curation, crowdsourcing, (semi-)automatic extraction, and/or a combination of those. Manually (i.e., expert) curated knowledge graphs, such as OpenCyc [32], can reach very high levels of accuracy, but only a limited coverage. On the other end of the spectrum, automatically created knowledge graphs, such as DBpedia [31], YAGO [33], or NELL [4], can reach a larger scale, but at a lower level of accuracy [39]. Table 10 depicts the most popular publicly available knowledge graphs and their respective sizes.

DBpedia and YAGO are created with Wikipedia as their input source. Both extract relations from infoboxes in Wikipedia using mappings from keys used in infoboxes to properties defined in a central ontology—for DBpedia, those mappings are crowdsourced [46], for YAGO, they are created by experts [33]. Furthermore, the DBpedia ontology defines a manually crafted class hierarchy, whereas YAGO creates a class hierarchy by combining Wikipedia’s category system with WordNet [12]. By design, both DBpedia and YAGO have a similar set of instances, i.e., each instance corresponds to a Wikipedia page. Wikidata also extracts information from Wikipedia, combined from different language editions, and exploits further external sources, such as library databases [53].

In contrast to this extraction, NELL is extracted from text from the Web. It uses a small set of seed facts and text patterns to iteratively learn new facts and patterns based on a Web crawl. Likewise, the WebIsALOD dataset [20,50] extracts a large taxonomy, i.e., a set of hypernymy relations, from a large-scale Web crawl. The resulting graph is very large, but contains no other relations beyond hypernymy. Since both NELL and WebIsALOD are extracted from text, they suffer from issues in entity disambiguation, which does not exist for Wikipedia-based graphs by design [41].

20 page https://memory-alpha.fandom.com/wiki/Jean-Luc_Picard#External_links contains links to Wikipedia Memory Beta etc.
Table 10 Public cross-domain knowledge graphs and their size

| Knowledge graph | # Entities | # RDF triples | Avg. indegree | Avg. outdegree | # Classes | # Relations |
|-----------------|------------|---------------|---------------|---------------|-----------|-------------|
| OpenCyc         | 118,499    | 2,413,894     | 10.03         | 9.23          | 116,822   | 165         |
| NELL            | 1,974,297  | 3,402,971     | 5.33          | 1.25          | 290       | 1334        |
| YAGO3           | 5,130,031  | 1,435,808,056 | 9.83         | 41.25         | 30,765    | 11,053      |
| DBpedia         | 5,109,890  | 397,831,457   | 13.52         | 47.55         | 754       | 3555        |
| DBkWik          | 11,163,719 | 91,526,001    | 0.70          | 8.17          | 12,029    | 128,566     |
| Wikidata        | 44,077,901 | 1,633,309,138 | 9.83         | 41.25         | 30,765    | 11,053      |
| WebIsALOD       | 212,184,968| 400,533,808   | 3.72          | 3.31          | –         | 1           |

Besides publicly available knowledge graphs, companies such as Google, Microsoft, or Facebook also maintain their own, non-public knowledge graphs. One approach which is close to DBkWik is Google’s Knowledge Vault [7], which combines data extracted from various sources on the Web, such as text, tables, and page structure. However, Knowledge Vault (like Google’s knowledge graph) is only used internally in Google applications and cannot be accessed or downloaded directly.

Table 10 summarizes public cross-domain knowledge graphs. Among those, it is remarkable that DBkWik has more than twice as many instances as the Wikipedia-based knowledge graphs DBpedia and YAGO.

6 Discussion

The DBkWik dataset shows that it is possible to create a large-scale knowledge graph which also cares about tail (not well known) entities. Nevertheless, each step in the DBkWik workflow can be improved.

Wiki dump retrieval Although we have targeted one Wiki hosting platform for this prototype, i.e., Fandom, the creation of the knowledge graph does not need to end there. WikiApiary reports more than 20,000 public installations of MediaWiki, all of which could be processed and integrated by the framework introduced in this paper.

Extraction of information Besides using infoboxes, as for DBpedia and YAGO, there are various approaches exploiting different structures in Wikipedia for knowledge graph construction and/or refinement, such as tables [34], lists [43], categories [18,45], textual content [19,26], or page links [36]. As all of those structures can also be observed in other Wikis, those approaches would be useful extensions for DBkWik as well.

Linking The matching of the separate knowledge graphs into a unified knowledge graph is a non-trivial task. Due to the high number of Wikis, the matching (even for the schema) cannot be done manually. Nevertheless, a semiautomatic approach with some manual intervention might be a way to go. This is similar to the interactive track [44] at the ontology alignment evaluation initiative (OAEI).

---

21 The numbers are taken from [20] and [47]. Note that WebIsALOD does not distinguish classes and instances; therefore, we cannot count classes.

22 https://wikiapiary.com/wiki/Statistics.
Despite lexical methods, combined with text embeddings of the Wikis from which DBkWik was extracted, yield good results, there is a lot of room for improvement. Therefore, we have released the gold standards for the matching part and created a new track at the ontology alignment evaluation initiative (OAEI) to foster the development of better matching approaches for the task of knowledge graph matching.

**Fusion** Another problem we have currently not considered is conflict detection and resolution. If we find different statements about an entity, they might be either complementary (e.g., different movies a person has acted in) or conflicting (e.g., different birthdays of a person). Conflicts may arise, e.g., due to errors or outdated information, but they may also hint at wrong fusions (e.g., two persons with the same name, but different birthdays). In the past, approaches for dealing with conflicts have been shown to work for different language editions from Wikipedia [3], which could be transferred to the DBkWik knowledge graph as well.

**Ontology Creation** The current ontology is rather shallow, consisting of a rudimentary class hierarchy and some domain and range restrictions. A more formalized ontology could be created by means of ontology learning tools [30]. The learned axioms could then also help further refining the instance level, e.g., by assigning additional statements and/or identifying conflicting ones [51] and the matching. Moreover, the exploitation of highly formalized top level ontologies could improve the ontology creation and matching process, as well as the enrichment and cleaning on the instance level [42].

**Refinement** In the past, many approaches for refining knowledge graphs have been proposed [37]. In the course of this paper, we have included a few of those (i.e., subclass induction for type completion, and a light-weight version of SDType); there is a large potential to apply more of those operators. For example, in [19], we have shown that relation extraction from abstracts can create a substantial improvement for Wiki-based knowledge graphs. However, extending the approach to multiple abstracts per Wiki might be another issue.

7 Conclusion and outlook

In this paper, we have introduced the DBkWik knowledge graph. It is created by processing the dumps of different Wikis using the DBpedia extraction framework, followed by data fusion and schema enrichment. The resulting knowledge graph, although so far only using a subset of available Wikis, is in the order of magnitude of current public, cross-domain knowledge graphs and has been shown to be rather complementary to Wikipedia-based knowledge graphs such as DBpedia. For further improvement of the matching component, we released the gold standard for interwiki linking as an OAEI track.

In the future, we plan to improve each step in the DBkWik workflow. At first, we want to crawl the Web for dumps of MediaWiki installations and include them in our knowledge graph. The extraction of links should be improved because interwiki links are not extracted at all in DBpedia. For interlinking the Wikis, we expect some good systems in the future participating at the OAEI campaign. These are well evaluated, and the best system might be used to create the fused knowledge graph. For DBkWik, there is a pipeline of several interdependent steps—e.g., schema matching, instance matching, data fusion, and refinement operators—and a systematic analysis of those interdependencies is still to be performed. Furthermore, joint approaches performing several of those steps simultaneously might be worth investigating.

Overall, we conclude that DBkWik is not only a novel cross-domain knowledge graph complementary to commonly known graphs such as DBpedia and YAGO, but also an inter-
est new testbed for novel methods for knowledge graph construction, refinement, and fusion.

Acknowledgements  We would like to thank Alexandra Hofmann, Samresh Perchani, and Jan Portisch, who helped developing the first prototype of DBkWik in the course of a student project.

References

1. Algergawy A, Cheatham M, Faria D, Ferrara I, Fundulaki I, Harrow I, Hertling S, Jiménez-Ruiz E, Karam N, Khiat A, Lambrix P, Li H, Montanelli S, Paulheim H, Pesquita C, Saveta T, Schmidt D, Shvaiko P, Splendidiani A, Thieblin E, Trojahn C, Vataščinová J, Zamazal O, Zhou L (2018) Results of the ontology alignment evaluation initiative 2018. In: OM 2018-13th ISWC workshop on ontology matching
2. Alstott J, Bullmore E, Plenz D (2014) Powerlaw: a Python package for analysis of heavy-tailed distributions. PloS one 9(1):e85777
3. Bryl V, Bizer C (2014) Learning conflict resolution strategies for cross-language Wikipedia data fusion. In: Proceedings of the 23rd international conference on world wide web. ACM, pp 1129–1134
4. Carlson A, Betteridge J, Wang RC, Hruschka Jr ER, Mitchell TM (2010) Coupled semi-supervised learning for information extraction. In: Proceedings of the third ACM international conference on web search and data mining, pp 101–110
5. Clauset A, Shalizi CR, Newman ME (2009) Power-law distributions in empirical data. SIAM Rev 51(4):661–703
6. Dohrn H, Riehle D (2011) Design and implementation of the Sweble Wikitext parser: unlocking the structured data of wikipedia. In: Proceedings of the 7th international symposium on wikis and open collaboration. ACM, pp 72–81
7. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun S, Zhang W (2014) 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. ACM, pp 601–610
8. Endris KM, Giménez-Garcia JM, Thakkai H, Demidova E, Zimmermann A, Lange C, Simperl E (2017) Dataset reuse: an analysis of references in community discussions, publications and data. Extraction 500:1
9. Erling O (2012) Virtuoso, a hybrid rdbms/graph column store. IEEE Data Eng Bull 35(1):3–8
10. Euzenat J, Meilicke C, Stuckenschmidt H, Shvaiko P, Trojahn C (2011) Ontology alignment evaluation initiative: six years of experience. J Data Semant XV:158–192
11. Faria D, Pesquita C, Balasubramani BS, Tervo T, Carriço D, Garrilha R, Couto FM, Cruz IF (2018) Results of AML participation in OAEI 2018. In: OM 2018-13th ISWC workshop on ontology matching
12. Fellebaum C (1998) WordNet—an electronic lexical database. MIT Press, Cambridge
13. Fleiss JL (1971) Measuring nominal scale agreement among many raters. Psychol Bull 76(5):378
14. Galárraga L, Teflioudi C, Hose K, Suchanek FM (2015) Fast rule mining in ontological knowledge bases with AMIE++. VLDB J Int J Very Large Data Bases 24(6):707–730
15. Guziewicz P, Manolescu I (2018) Quotient RDF summaries based on type hierarchies. In: DESWeb 2018—data engineering meets the semantic web 2018
16. Hauser DJ, Schwarz N (2016) Attentive Turkers: MTurk participants perform better on online attention checks than do subject pool participants. Behav Res Methods 48(1):400–407. https://doi.org/10.3758/s13428-015-0578-z
17. Heath T, Bizer C (2011) Linked data: evolving the web into a global data space, vol 1, no 1. Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool, San Rafael, pp 1–136
18. Heist N, Paulheim H (2019) Uncovering the semantics of Wikipedia categories. In: International semantic web conference
19. Heist N, Hertling S, Paulheim H (2018) Language-agnostic relation extraction from abstracts in Wikis. Information 9(4):75
20. Hertling S, Paulheim H (2017) Webisalod: providing hypernymy relations extracted from the web as linked open data. In: International semantic web conference. Springer, pp 111–119
21. Hertling S, Paulheim H (2018a) Dbkwik: A consolidated knowledge graph from thousands of wikis. In: 2018 IEEE international conference on big knowledge (ICBK). IEEE, pp 17–24
22. Hertling S, Paulheim H (2018b) Dome results for OAEI 2018. In: OM 2018-13th ISWC workshop on ontology matching
23. Hofmann A, Perchani S, Portisch J, Hertling S, Paulheim H (2017) Dbkwik: towards knowledge graph creation from thousands of wikis. In: International semantic web conference (posters and demos)
Publisher's Note  Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Sven Hertling is a Ph.D. student at the Data and Web Science Group, University of Mannheim, Germany. His research focuses on generating large-scale knowledge graphs and especially on their integration. He is a co-organizer of the knowledge graph matching track at the ontology alignment evaluation initiative (OAEI). Previously, he worked in the Smart Data & Knowledge Services group of the German Research Center for Artificial Intelligence in Kaiserslautern, focusing on applications of semantic web technologies in industrial settings. Sven holds a B.Sc. and M.Sc. degree from the Technical University of Darmstadt, Germany.

Heiko Paulheim is a professor for Data Science at the University of Mannheim, Germany. His research focuses on the construction and usage of large-scale knowledge graphs. He explores methods for generating those knowledge graphs from various sources (such as Wikis and other structured Web sites or parts thereof), as well as techniques for automatically refining those graphs by inferring missing knowledge or finding errors by means of heuristic inference or machine learning. Moreover, Heiko takes a holistic view on knowledge graph construction and refinement by trying to explicate and leverage meta-knowledge on the knowledge construction process and lifecycle. From an application perspective, he analyzes and explores the usage of knowledge graphs to improve the performance of different knowledge-intensive tasks. Here, he has developed methods for exploiting public knowledge graphs as background knowledge in data mining tasks, both using symbolic and subsymbolic methods. Heiko holds a Ph.D. from the Technical University of Darmstadt, Germany.