DBkWik: A Consolidated Knowledge Graph from Thousands of Wikis

Sven Hertling
Data and Web Science Group
University of Mannheim
Mannheim, Germany
sven@informatik.uni-mannheim.de

Heiko Paulheim
Data and Web Science Group
University of Mannheim
Mannheim, Germany
heiko@informatik.uni-mannheim.de

***Abstract*—Popular 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.**

***Index Terms*—Knowledge Graph Creation, Information Extraction, Linked Open Data**

I. Introduction

General purpose knowledge graphs such as DBpedia, YAGO, and Wikidata, have become a central part of the Linked Open Data cloud [1] and are among the most frequently used datasets within the Web of data [2]. Such knowledge graphs contain information on millions of entities from multiple topical domains [3].

Many of the popular knowledge graphs are created from Wikipedia, and hence have a similar coverage [4]. 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 [5].

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.

The rest of this paper is structured as follows. Section II describes the approach for creating DBkWik. Section III gives insights into the topology and contents of the resulting knowledge graph. We close with a review of related work and an outlook on future developments.

II. Approach

For creating the DBkWik knowledge graph, we use the 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[5] as input and produces a knowledge graph as output. 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 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 as well as the mappings are created manually in a collaborative workflow.

Since neither manually created mappings nor a central ontology exist 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

[1] pronounced *dee-bee-quick*
[2] http://dbkwik.webdatacommons.org/
[3] http://creativecommons.org/licenses/by-sa/3.0/
[4] https://github.com/dbpedia/extraction-framework

[5] 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.

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, this property is not given, hence, duplicate detection (i.e., matching the knowledge graphs extracted from the individual Wikis to each other) and data fusion must be performed in addition to the extraction.

In addition to the pure extraction and fusion, we also apply two knowledge graph refinement operations, i.e., type induction using a simplified version of *SDType* [6], and schema enrichment including subclass relations as well as domain and range restrictions. The final dataset is loaded into a Virtuoso server [7], which makes the knowledge graph available both as a Linked Data service as well as a SPARQL endpoint. Fig. 1 shows the overall workflow of the DBkWik generation.

A. 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[6] 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 towards 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
- All mapping-based extractors were removed (cf. a)
- Types are automatically created from infobox template names instead (cf. a)
- Abstracts are extracted using the Sweble parser [8] 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.

B. Linking to DBpedia

To make DBkWik a five star dataset [9] and a proper citizen of the Linked Open Data cloud [1], 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 F1 score above 0.85 [10].

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 [11]. We train a doc2vec model [12] 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 [12], 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.

C. 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* [13] from the original extraction, given both an equivalence relation for the schema and instance level. 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. Fig. 2 illustrates that process.

D. 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 [14], using association rule mining for determining both subclass relations as well as domain and range restrictions. Following

[6]https://github.com/sven-h/extraction-framework

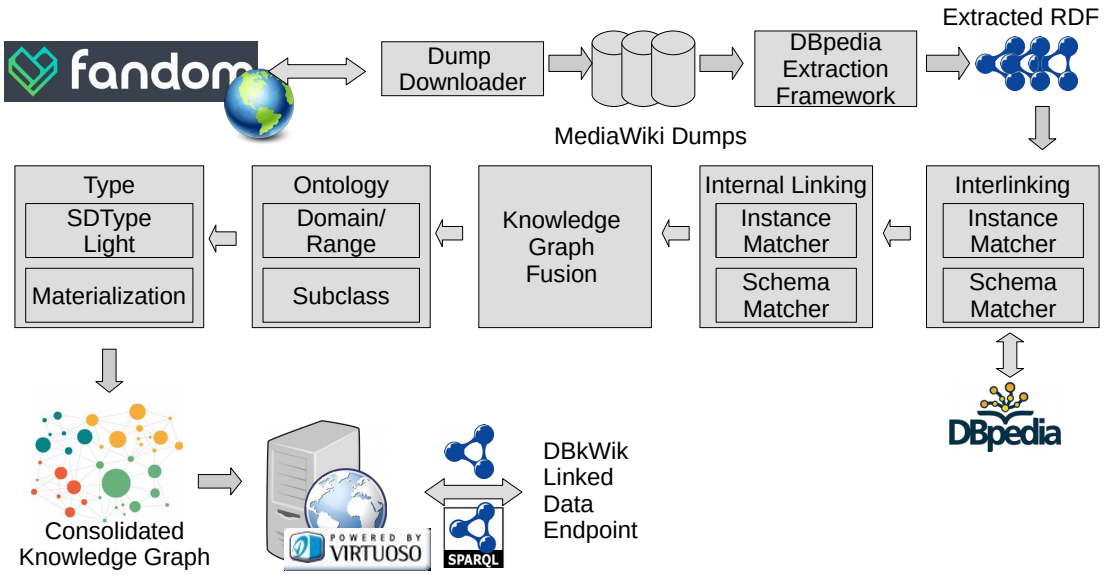


Fig. 1: The overall workflow creating the DBkWik knowledge graph

the observation that in a skewed dataset like a general purpose knowledge graph, a minimum support threshold is not very meaningful [15], 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

$$\texttt{dbkwik:Artist} \rightarrow \texttt{dbkwik:Person}$$

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

```
dbkwik:87a0c82f rdf:type dbkwik:Person .
```

The domain and range restrictions are used to implement a light-weight version *SDType* [6], 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.,

$$SDType(o,c) = \nu \cdot \sum_{(s,p,o)\in G} w_p \cdot P((o,type,c) \in G \mid (x,p,o) \in G),$$

using a normalization factor ν and some property weights w_p, we use the inferred ranges instead for a simplified and more efficient computation:

$$SDLight(o) = \underset{c \in classes(G)}{\arg\max} |\{p|(s,p,o) \in G \wedge (p,range,c) \in G\}|$$

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.

III. Analysis of the Resulting Dataset

Following the methodology described above, we applied the approach to *Fandom powered by Wikia*[7], which is one of the most popular *Wiki Farms*[8], comprising more than 423,000 individual Wikis totaling nearly 40 million articles.

A. 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.

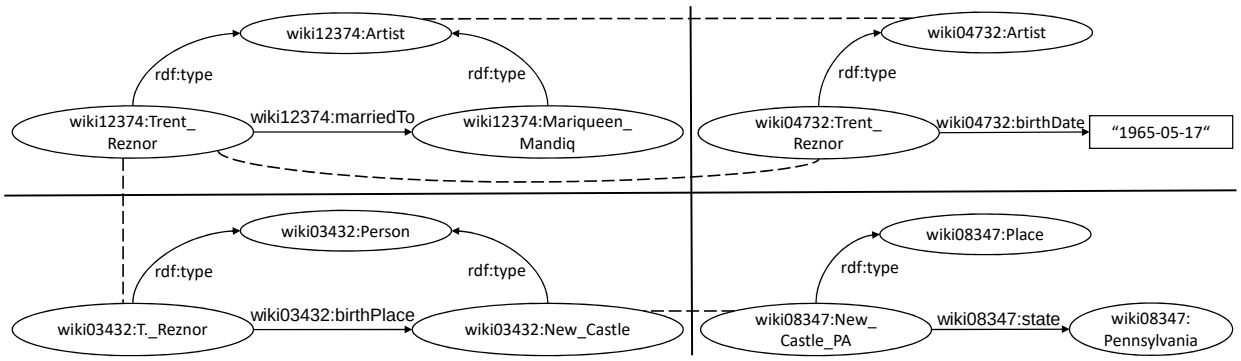
B. The DBkWik Knowledge Graph

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 I depicts the characteristics of that initial knowledge graph extracted (column *initial*).

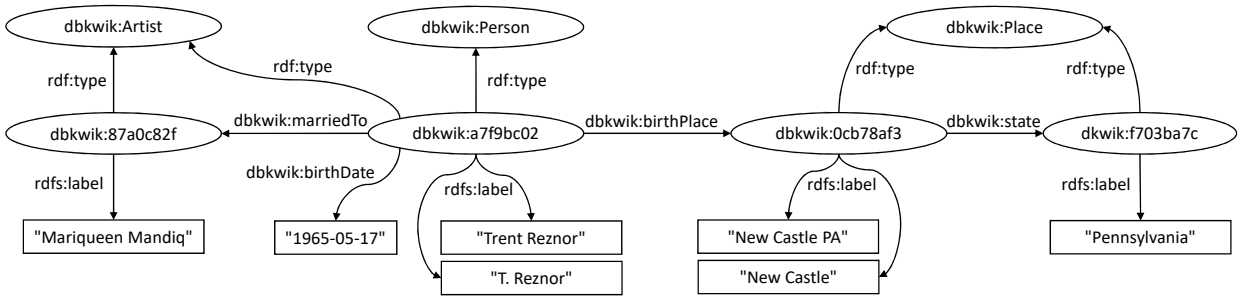
The internal matching and fusion reduces the number of instances by a factor of 21.4%, and also the number of statements by a factor of 15.1%. Note that the latter factor is smaller, because a statement `s p o` is removed if and only if there is a statement `s' p' o'`, where all three of

[7] http://fandom.wikia.com/

[8] http://www.alexa.com/topsites/category/Computers/Software/Groupware/Wiki/Wiki_Farms



(a) Knowledge graphs from four different Wikis.



(b) Fused knowledge graph.

Fig. 2: Example for the internal linking and fusion, depicting the state before the fusion, with interlinks as dashed lines (top), and after the fusion (bottom).

TABLE I: 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

the following apply: `s` is mapped to `s'`, `p` is mapped to `p'`, and `o` is mapped to `o'`.

Table II 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: 3,391).

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*[9] in table II. This is due to the fact that the textual content of the Wiki pages is used as the main signal for interlinking, without taking cross-lingual matching into account.

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. In total, materializing the subclass relations leads to 626 additional `rdf:type` statements. After applying the lightweight type induction based on SDType, we obtain another 97,293 `rdf:type` statements to the knowledge graph.

The characteristics of the knowledge graph after the fusion and enrichment steps. We can observe that while the number of instances and statements is reduced, the average degree and the fraction of typed instances increase, i.e., the knowledge graph gets smaller and more densely connected.

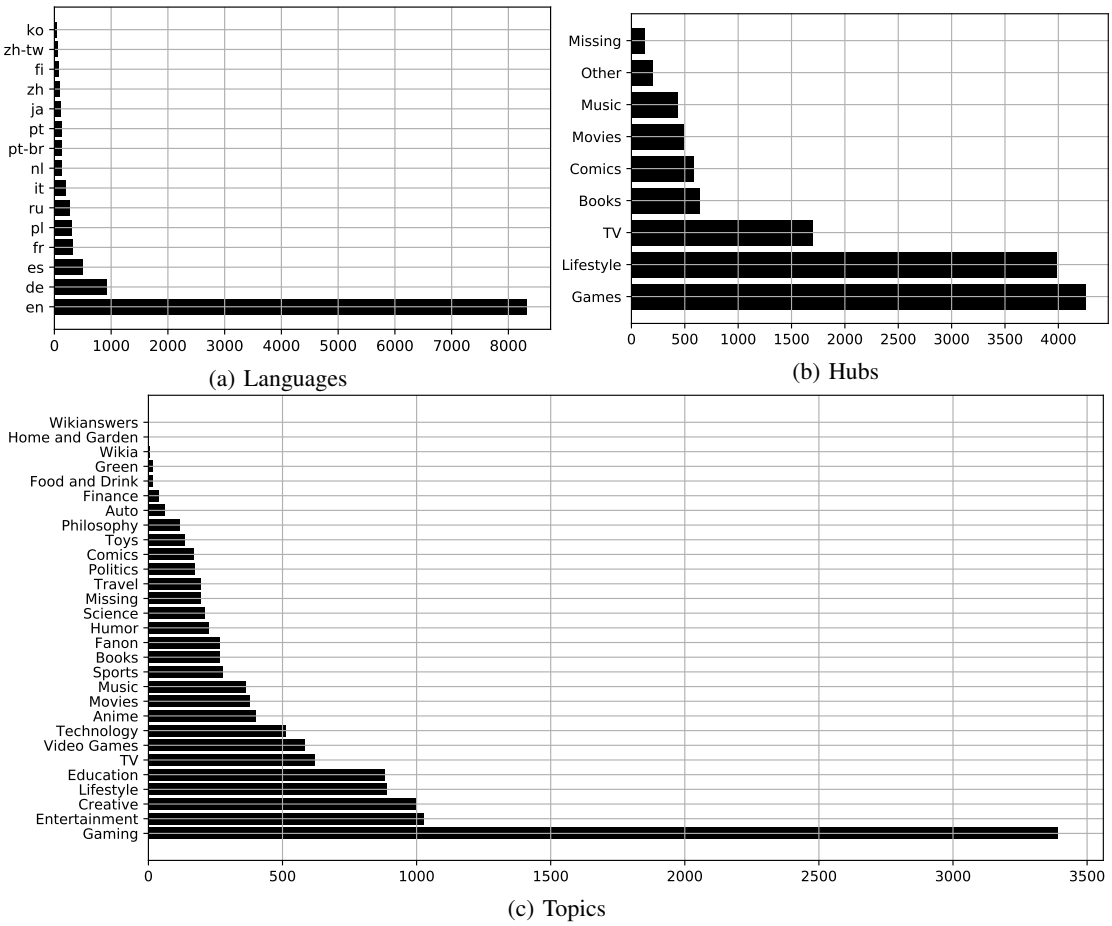
C. Linkage Quality

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 [10] 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 are noticed[10].

The resulting interlinking quality is shown in table III. For the four approaches using doc2vec, we also depict the optimal

[9] *Hauptseite* is German for *main page*.

[10] The gold standard is available at https://github.com/sven-h/dbkwik



(a) Languages

(b) Hubs

(c) Topics

Fig. 3: Breakdown of the downloaded Wikis by Language, Topic, and Hub

TABLE II: Most frequently matched classes, properties, and instances

Classes		Properties		Instances	
name	#match	name	#match	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

threshold. It can be observed that the schema matching based on simple string matching yields good results, and that using doc2vec for instance matching brings a significant advantage.

For evaluating the internal interlinking, we created a gold standard using a dual approach. 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 (gaming, comics, and entertainment) 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 5 crowd workers.[11] The inter-rater agreement according to Fleiss' kappa [18] is 0.8762, which is an *almost perfect agreement* according to [19].

The results of the evaluation of the internal interlinking are shown in table IV, 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

[11] We restricted the workers to have a 95% approval rate and a minimum of 100 approved HITs (human intelligence tasks), following the recommendations by [16] and [17], and restricted their location to the US 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-h/dbkwik.

TABLE III: Performance of interlinking to DBpedia

		Macro avg.			Micro avg.		
		P	R	F1	P	R	F1
Classes		1.000	.831	.908	1.000	.815	.898
Properties		.889	.843	.866	.890	.841	.865
Instances	String similarity	.459	.676	.547	.449	.657	.533
	PV-DM (short) t=.143	.469	.657	.547	.468	.657	.547
	PV-DM (long) t=.203	.493	.705	.581	.495	.701	.580
	PV-DBOW (short) t=.416	.537	.657	.591	.538	.642	.585
	PV-DBOW (long) t=0.5	.663	.689	.676	.643	.672	.657

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.

D. Complementarity to DBpedia

In [4], 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 according to that overlap is 528,458. In other words, only 95.3% of all entities in DBkWik are contained *exclusively* in DBkWik and not in DBpedia. Likewise, 89.7% of all entities in DBpedia are not 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 V 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 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.

IV. 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 [20], 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 [21], YAGO [22], or NELL [23], can reach a larger scale, but at a lower level of accuracy. Table VI 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 [24], for YAGO, they are created by experts [22]. Furthermore, the DBpedia ontology defines a manually crafted class hierarchy, whereas YAGO creates a class hierarchy by combining Wikipedia's category system with WordNet [25]. 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 [26].

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 [27], [28] 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 [29].

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* [30], 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 VI summarizes public cross-domain knowledge graphs.[12] Among those, it is remarkable that DBkWik has more than twice as many instances as the Wikipedia-based knowledge graphs DBpedia and YAGO.

[12]The numbers are taken from [4] and [27]. Note that WebIsALOD does not distinguish classes and instances, therefore, we cannot count classes.

TABLE IV: Performance of internal interlinking within DBkWik

		Macro avg.			Micro avg.		
		P	R	F1	P	R	F1
Classes		.990	.990	0.990	.979	.979	.979
Properties		.840	.809	.824	.860	.813	.836
Instances	String similarity	.919	.824	.869	.920	.846	.881
	PV-DM (short) t=.143	.919	.824	.869	.920	.845	.881
	PV-DM (long) t=.203	.919	.824	.869	.920	.845	.881
	PV-DBOW (short) t=.416	.921	.824	.870	.924	.846	.883
	PV-DBOW (long) t=0.5	.921	.816	.866	.923	.838	.879

TABLE V: The largest classes in DBkWik with instance counts in DBkWik and DBpedia (– indicates that no corresponding class exists in 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	9,225
Actor	20,278	6,697
Ship	18,591	35,486
Jutsu	18,122	–
Military Person	17,577	33,181
Quest	17,272	–

V. 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.

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[13], all of which could be processed and integrated by the framework introduced in this paper. In the future, we plan to crawl the Web for dumps of MediaWiki installations and include them in our knowledge graph.

In the past, many approaches for refining knowledge graphs have been proposed [3]. 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 [11], 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.

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. In the past, approaches for dealing with conflicts have been shown to work for different language editions from Wikipedia [31], which could be transferred to the DBkWik knowledge graph as well.

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.

Since there are many interesting opportunities for contributing to the quality of DBkWik, we have not only released the dataset, but also the corresponding gold standards, e.g., for schema and instance matching. The latter is also used in a novel knowledge graph matching task at the annual Ontology Alignment Evaluation Initiative (OAEI).[14]

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 interesting new testbed for novel methods for knowledge graph construction 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] M. Schmachtenberg, C. Bizer, and H. Paulheim, "Adoption of the linked data best practices in different topical domains," in *International Semantic Web Conference*. Springer, 2014, pp. 245–260.

[2] K. M. Endris, J. M. Giménez-García, H. Thakkar, E. Demidova, A. Zimmermann, C. Lange, and E. Simperl, "Dataset reuse: An analysis of references in community discussions, publications and data," *Extraction*, vol. 500, p. 1, 2017.

[3] H. Paulheim, "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods," *Semantic Web*, 2016.

[4] D. Ringler and H. Paulheim, "One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co." in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 2017, pp. 366–372.

[5] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio, "Sprank: Semantic path-based ranking for top-n recommendations using linked open data," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 9, 2016.

[6] H. Paulheim and C. Bizer, "Type inference on noisy rdf data," in *International Semantic Web Conference*. Springer, 2013, pp. 510–525.

[13] https://wikiapiary.com/wiki/Statistics

[14] http://oaei.ontologymatching.org/2018/knowledgegraph/

TABLE VI: Public cross-domain knowledge graphs and their size

Knowledge Graph	# Entites	# 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	1,334
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	3,555
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

[7] O. Erling, "Virtuoso, a hybrid rdbms/graph column store." *IEEE Data Eng. Bull.*, vol. 35, no. 1, pp. 3–8, 2012.

[8] H. Dohrn and D. Riehle, "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, 2011, pp. 72–81.

[9] T. Heath and C. Bizer, "Linked data: Evolving the web into a global data space," *Synthesis lectures on the semantic web: theory and technology*, vol. 1, no. 1, pp. 1–136, 2011.

[10] A. Hofmann, S. Perchani, J. Portisch, S. Hertling, and H. Paulheim, "Dbkwik: towards knowledge graph creation from thousands of wikis," in *International Semantic Web Conference (Posters and Demos)*, 2017.

[11] N. Heist, S. Hertling, and H. Paulheim, "Language-agnostic relation extraction from abstracts in wikis," *Information*, vol. 9, no. 4, p. 75, 2018.

[12] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[13] P. Guzewicz and I. Manolescu, "Quotient rdf summaries based on type hierarchies," in *DESWeb 2018 - Data Engineering meets the Semantic Web 2018*, 2018.

[14] J. Völker and M. Niepert, "Statistical schema induction," in *Extended Semantic Web Conference*. Springer, 2011, pp. 124–138.

[15] H. Paulheim, "Data-driven joint debugging of the dbpedia mappings and ontology," in *European Semantic Web Conference*. Springer, 2017, pp. 404–418.

[16] G. Kazai, *In Search of Quality in Crowdsourcing for Search Engine Evaluation*. Springer Berlin Heidelberg, 2011, pp. 165–176.

[17] D. J. Hauser and N. Schwarz, "Attentive turkers: Mturk participants perform better on online attention checks than do subject pool participants," *Behavior Research Methods*, vol. 48, no. 1, pp. 400–407, 2016.

[18] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[19] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[20] D. B. Lenat, "CYC: A large-scale investment in knowledge infrastructure," *Communications of the ACM*, vol. 38, no. 11, pp. 33–38, 1995.

[21] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia," *Semantic Web Journal*, vol. 6, no. 2, 2013.

[22] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *CIDR*, 2013.

[23] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 101–110.

[24] M. Rico, N. Mihindukulasooriya, D. Kontokostas, H. Paulheim, S. Hellmann, and A. Gómez-Pérez, "Predicting incorrect mappings : a data-driven approach applied to dbpedia," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 323–330.

[25] C. Fellbaum, *WordNet – An Electronic Lexical Database*. MIT Press, 1998.

[26] D. Vrandečić and M. Krötzsch, "Wikidata: a Free Collaborative Knowledge Base," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[27] S. Hertling and H. Paulheim, "Webisalod: providing hypernymy relations extracted from the web as linked open data," in *International Semantic Web Conference*. Springer, 2017, pp. 111–119.

[28] J. Seitner, C. Bizer, K. Eckert, S. Faralli, R. Meusel, H. Paulheim, and S. P. Ponzetto, "A large database of hypernymy relations extracted from the web." in *LREC*, 2016.

[29] H. Paulheim and C. Bizer, "Improving the quality of linked data using statistical distributions," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 2, pp. 63–86, 2014.

[30] X. 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*. ACM, 2014, pp. 601–610.

[31] V. Bryl and C. Bizer, "Learning conflict resolution strategies for cross-language wikipedia data fusion," in *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 2014, pp. 1129–1134.