Wikipedia Knowledge Graph for Explainable AI*

Md Kamruzzaman Sarker[1], Joshua Schwartz[1], Pascal Hitzler[1], Lu Zhou[1], Srikanth Nadella[3], Brandon Minnery[3], Ion Juvina[2], Michael L. Raymer[2,3], and William R. Aue[2]

[1] Kansas State University, Manhattan, KS 66506, USA
[2] Wright State University, Dayton, OH 45435, USA
[3] Kairos Research, Dayton, OH 45458, USA

Abstract. Explainable artificial intelligence (XAI) requires domain information to explain a system's decisions, for which structured forms of domain information like Knowledge Graphs (KGs) or ontologies are best suited. As such, readily available KGs are important to accelerate progress in XAI. To facilitate the advancement of XAI, we present the cycle-free Wikipedia Knowledge Graph (WKG) based on information from English Wikipedia. Each Wikipedia article title, its corresponding category, and the category hierarchy are transformed into different entities in the knowledge graph. Along with cycle-free version we also provide the original knowledge graph as it is. We evaluate whether the WKG is helpful to improve XAI compared with existing KGs, finding that WKG is better suited than the current state of the art. We also compare the cycle-free WKG with the Suggested Upper Merged Ontology (SUMO) and DBpedia schema KGs, finding minimal to no information loss.

Keywords: Knowledge Graph · Wikipedia · Ontology · XAI

1 Introduction

Artificial intelligence (AI)—including the subfields of machine learning and deep learning—has advanced considerably in recent years. In tandem with these performance improvements, understanding how AI systems make decisions has become increasingly difficult due to many nonlinear transformations of input data and the complex nature of the algorithms involved. The research area explainable AI (XAI) [8,7,16] investigates techniques to examine these decision processes.

A main desideratum of XAI is user understandability [6,5], while explanations should take into account the context of the problem and relevant domain knowledge [10]. Humans understand and reason mostly in terms of concepts and combinations thereof. A knowledge graph (KG) embodies such understanding in links between concepts; such a natural conceptual network creates a pathway to use knowledge graphs in XAI applications to improve overall understandability of complex AI algorithms. For an overview of some of the current discussion on

* This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111890019

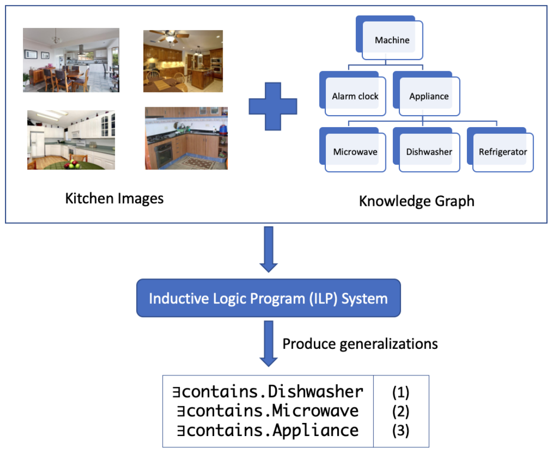


Fig. 1: Example of using knowledge graph to enhance explainability

utilizing knowledge graphs to enhance explanations, and possible limitations of existing approaches, see [12,9].

One of the primary elements of knowledge graphs to use in the XAI context is the notion of a concept hierarchy [4,18]. As illustrated in Figure 1, consider a system trying to explain the decisions of an image classifier. It may determine that an image should be given the label "Kitchen" because it contains a dishwasher, refrigerator, and microwave, and with the help of a KG concept hierarchy, it may produce the more general explanation that the image contains items in the "Appliance" class. These kinds of explanation generation systems are based on inductive logic programming (ILP) [14], and rich concept hierarchies play an important role in the generation of satisfactory explanations. To advance the state of XAI research, we provide a readily available knowledge graph with a rich concept hierarchy.

Wikipedia is perhaps the largest high-quality free source of information on the web. Wikipedia articles are classified into human-managed categories, which form a hierarchy (albeit with cycles). These concepts embody humans' natural ways of thinking and are easily understood, providing a greater benefit in an XAI context.

DBpedia [1], Suggested Upper Merged Ontology (SUMO) [15], Freebase [2], and Yago [19] are among the many high-quality, publicly available knowledge graphs providing domain information. These KGs use information from many sources, including Wikipedia. The hierarchical category information of Wikipedia, in which we are interested, is available in SUMO[1] but not in Freebase. It also exists in DBpedia and is accessible through SPARQL queries. Problematically, though, the Wikipedia parts of SUMO and the DBpedia KG contain cycles. For example, consider the following two axioms from DBpedia.

[1] `http://www.adampease.org/OP/`

I. *1949_establishments_in_Asia skos:broader 1949_establishments_in_India*
II. *1949_establishments_in_India skos:broader 1949_establishments_in_Asia*

These axioms form a cycle in the Wikipedia category hierarchy and hence also in DBpedia. The Wikipedia category hierarchy contains many such cycles, which complicates its use in XAI applications, as choosing parent concepts from the KG becomes nondeterministic.

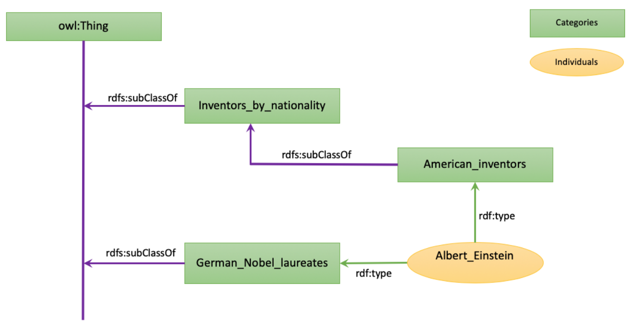


Fig. 2: Example architecture of the Wikipedia knowledge graph

To solve this problem, we provide a noncyclic version of the Wikipedia category hierarchy knowledge graph. We also empirically evaluate how the noncyclic knowledge graph performs in an XAI context and whether breaking cycles degrades its quality, finding that the Wikipedia knowledge graph performs better in both scenarios than other existing knowledge graphs.

The rest of the paper is organized as follows. First, we describe the high level architecture of the knowledge graph in section 2. Next, we describe the steps involved in building the knowledge graph. Then, in section 4, we evaluate the knowledge graph before concluding.

2 Knowledge Graph Architecture

We want to make the knowledge graph as simple as possible to enable use within XAI applications with minimal preprocessing. In the knowledge graph, we will have entities (named individuals in OWL 2), their types (classes in OWL 2), and the types' hierarchy. Many relations can be extracted from Wikipedia, but for simplicity we will use only two: *rdf:type* and *rdfs:subClassOf*. The relation *rdf:type* will be used to assign the individuals to their corresponding types, and the *rdfs:subClassOf* relation will be used to create the hierarchy. The title of a Wikipedia article (a.k.a. page) becomes an entity in our KG. Categories of a page become the types of the corresponding individual. A subcategory relationship becomes a *rdfs:subClassOf* relationship.

Figure 2 shows the architecture of our knowledge graph with an example. We can see that the article *Albert_Einstein* is mapped into the knowledge graph as an individual. This article belongs to many categories, including *German_Nobel_laureates* and *American_inventors*, which are converted into instances of *rdfs:Class*. The category *American_inventors* is a subcategory of *Inventors_by_nationality*, among others, resulting in the relation

American_inventors rdfs:subClassOf Inventors_by_nationality

in the KG.

3 Generating the Knowledge Graph

We now briefly describe a procedure for generating a knowledge graph like the one discussed above from the version of Wikipedia for a particular language; full details are in Appendix A. To construct the Wikipedia category hierarchy knowledge graph from scratch, we explored two alternative approaches: traversing and parsing the hierarchy page by page, and using a Wikipedia data dump.[2] To get all page and category information from Wikipedia through a traversal, we

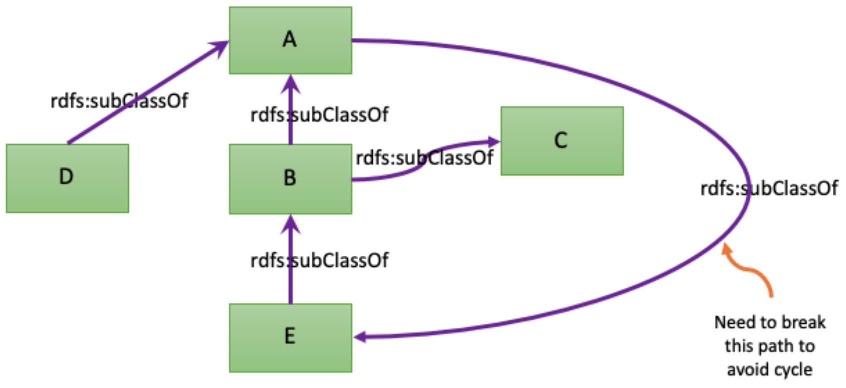


Fig. 3: Example of how cycles are broken

start at the top category[3] and exhaustively look through its subcategories and pages recursively, a time-consuming process complicated by the need to parse each page to find the proper links to visit the next categories or pages. To determine how long this process takes in practice, we used Python to implement the

[2] `http://dumps.wikimedia.org/enwiki/latest`
[3] `https://en.wikipedia.org/wiki/Category:Main_topic_classifications`

visiting and scraping program and found that it took roughly five days on a 2.2 GHz Intel Core i5 machine with 32 GB memory. As taking five days to produce a knowledge graph is not reasonable, we will focus on the Wikipedia data dump option.

A Wikipedia data dump contains all the information for each article: full text, editor list, category, etc. As stated in Section 2, our knowledge graph includes article title, category name, and the hierarchy of categories. These data are stored in the *page* and *categorylinks* tables. Using the Wikipedia data dump is straightforward: we just need to download the dump, import it into a database, and access it through SQL queries. After importing it, producing the full knowledge graph took only one hour, on the order of 1% of the time of the previous approach.

3.1 Concrete Implementation

Following the steps mentioned in Appendix A, we can create a concrete Wikipedia knowledge graph, ensuring compliance with W3C standards to make it maintainable, reusable, and non-proprietary. Many tools are available for this; among the most popular are the OWL API [11], the Apache Jena[4] library, and Owlready2,[5], all of which are compliant with W3C's standards.

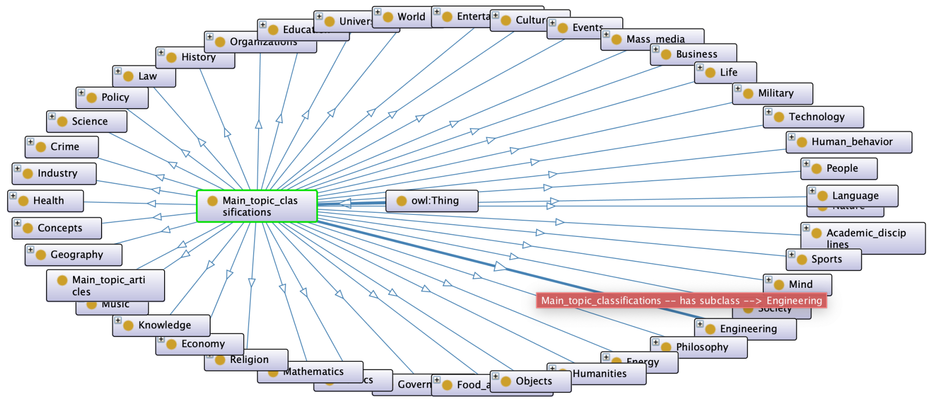


Fig. 4: Wikipedia Knowledge Graph

As discussed in Section 1, the raw Wikipedia hierarchy has cycles, resulting in cyclic relations in the knowledge graph. The Owlready2 library treats concepts as Python classes, representing subclass relationships through inheritance; since Python only supports inheritance without cycles, Owlready2 cannot handle these

[4] https://jena.apache.org/
[5] https://pythonhosted.org/Owlready2/

Table 1: Entity counts for Wikipedia, SUMO, and DBpedia knowledge graphs

Number of entities/facts	SUMO	DBpedia	Wikipedia cyclic	Wikipedia noncyclic
Concepts	4558	1183	1,901,708	1,860,342
Individuals	86,475	1	6,145,050	6,079,748
Object property	778	1144	2	2
Data property	0	1769	0	0
Axioms	175,208	7228	71,344,252	39,905,216
Class assertion axioms	167381	1	57,335,031	27,991,282
Subclass axioms	5330	769	5,962,463	3,973,845

cycles in relations. In contrast, the OWL API and Jena can support these cyclic relations; we use the former.[6]

While making the KG we face some practical issues, one being that many page titles on Wikipedia have non-ASCII characters, multiple spaces, and other peculiarities. For example, the article `https://en.wikipedia.org/wiki/Polish_People%27s_Party_%22Piast%22_(1913%E2%80%931931)` has title Polish_People%27s_Party_%22Piast%22_(1913%E2%80%931931). From an ontological perspective, this title as an entity name seems bad. We decide to replace spaces and characters in the set

```
`~!@#$%^&*()-+={}[]|\;'"<>,.?/
```

with underscores (_) and then trim leading and trailing underscores from the resulting string. Another technical issue consists in the fact that if proper Unicode rendering is not selected, some article names will be saved as non–Unicode-compliant names. For example, as of 20 January 2020, the article title *Fabian's Lizard* contains the additional character *0x92* just before the s. This character only exists in windows encoding cp1252 and not in Unicode.[7]

3.2 Breaking Cycles

As stated above, the Wikipedia category hierarchy contains cycles, which we break by visiting the categories using breadth-first search (BFS). Starting from the root—*Main topic classi cations*—we go level by level. An example of breaking a cycle is shown in Figure 3. In the example, if we start from A using BFS, we will get B and D as subclasses of A. On the next level, starting from B, we see that E is a subclass of B and store that information. On the next level, starting at E, we see that A is subclass of E; this results in a cycle, so we discard this information. Breaking cycles in this way results in some missing information in the final graph; however, it simplifies the knowledge graph considerably, allowing for efficient parent category determination, which is especially helpful in the XAI context.

[6] Our code is available at `https://github.com/md-k-sarker/Wiki-KG`.
[7] `https://stackoverflow.com/q/29419322/1054358`

Entity counts for both the cyclic and noncyclic versions of the WKG are shown in table 1. We see that breaking cycles results in losing 41,366 concepts (0.02% of the total 1,901,708 concepts) and 65,302 individuals (0.01% of the total 6,145,050 individuals). We further see that we lose a substantial number of class assertion axioms—29,341,749, or 0.5% of the total noncyclic axioms. Figure 4 shows a top-level view of the complete knowledge graph.[8]

4 Evaluation

The goal of our experimental evaluation was to test the hypothesis that the Wikipedia Knowledge graph produces XAI results comparable to or better than existing knowledge graphs. As to the best of our knowledge only SUMO has been used previously in a comparable context [18], to test this we compared the performance of our newly created WKG with that of the SUMO KG. We further hypothesized that breaking cycles in the Wikipedia knowledge graph results in minimal information loss and evaluated WKG relative to SUMO and the DBpedia schema.[9]

4.1 WKG's Effectiveness in XAI

To the best of our knowledge, there is no previously established quantitative measure of XAI quality, so we decided to use the accuracy metric of inductive logic programming (ILP)—the backbone of XAI [18]—to explain a supervised machine learning algorithm's decisions in terms of a KG. ILP provides many alternative solutions by using a KG. To measure a solution's performance, we used coverage score, described in equation (1), as the objective function. To measure the overall performance of a KG, we calculated the average of all scores of the produced solution for an experiment with equation (2).

$$Coverage(S) = \frac{P_S + N_{NS}}{P_S + P_{NS} + N_S + N_{NS}} \tag{1}$$

where

$$\begin{aligned} P_S &= \text{Number of positive individuals subsumed by the solution} \\ P_{NS} &= \text{Number of positive individuals not subsumed by the solution} \\ N_S &= \text{Number of negative individuals subsumed by the solution} \\ N_{NS} &= \text{Number of negative individuals not subsumed by the solution} \end{aligned}$$

$$Average\ coverage = \sum_{i=1}^{n} Coverage(S_i) \tag{2}$$

[8] Available for download at `https://osf.io/3wbyr/`.

[9] `http://downloads.dbpedia.org/2014/dbpedia_2014.owl.bz2`

Following [18], we used the ADE20K dataset [20], which contains over 20,000 images classified by scene type and annotated with contained objects, to compare the results. We cast the ADE20k dataset, with annotations, into an OWL ontology and aligned it with SUMO, as in [18]; in the present context, we also aligned the ontology with WKG. We use all five experiments mentioned in [18], but expand the range of the experiments. While the previous paper used only 3–10 images for each experiment, we took all the training images (around 100) of the relevant categories from the ADE20K dataset. To get the explanation, we use ECII [17] instead of DL-Learner [3] to avoid the latter's considerable time complexity.

Table 2: Comparison of average coverage for WKG and SUMO in XAI context

Experiment name	#Images	#Positive images	Wikipedia		SUMO	
			#Solution	Coverage	#Solution	Coverage
Market vs. WorkRoom and WareHouse	96	37	286	.72	240	.72
Mountain vs. Market and WorkRoom	181	85	195	.61	190	.53
OutdoorWarehouse vs. IndoorWarehouse	55	3	128	.94	102	.89
Warehouse vs. Workroom	59	55	268	.56	84	.24
Workroom vs. Warehouse	59	4	128	.93	93	.84

We will now briefly discuss each of the scenarios in turn, before we summarize; Table 2 Figure 5 provide an overview of the results.

The first experiment involved finding a generalization of market images from the market vs. workroom and warehouse images. The ADE20K training dataset has, for those three categories, a total of 96 images, all of which we used. The objective was to cover as *many* as possible of the 37 images of market scenes and as *few* as possible of the images of workroom and warehouse scenes. When using the Wikipedia knowledge graph, the explanation framework (ECII) produced 286 alternative rules to generalize the market images, while using the SUMO knowledge graph results in 240 alternative rules. Average coverage score for both Wikipedia and SUMO was 0.72, i.e. in this case the simple Wikipedia category hierarchy knowledge graph performs as well as SUMO.

To produce a generalized rule of mountain scenes was the objective of the second experiment. All 181 images from the ADE20K training set were taken in this mountain vs. market and workroom experiment, where 85 images were of mountain scenes. The average coverage for Wikipedia was 0.61, representing slightly better performance than the 0.53 coverage we obtained for SUMO.

In the ADE20K training data, only three images are of outdoor warehouse scenes, while 52 are of indoor warehouse scenes. We wanted to compare the performances of the WKG and SUMO given such skewed sizes of sets of positive and negative individuals, so we took the three images of outdoor warehouses and 52 images of indoor warehouses, aiming to produce a generalized rule to describe the outdoor warehouse scenes. As there are fewer images to describe, both SUMO and Wikipedia performed well: ECII produced average coverages

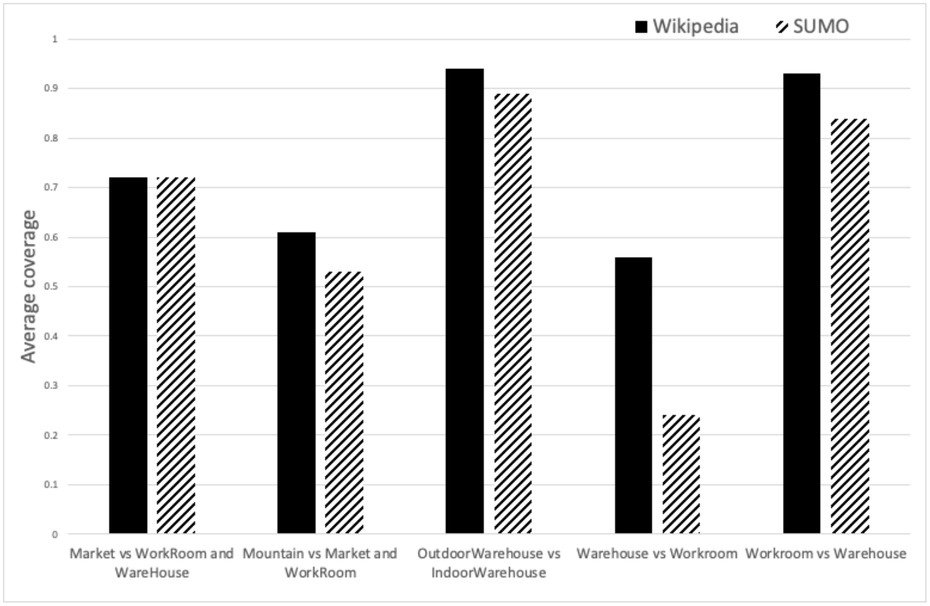


Fig. 5: Comparison of average coverage score between Wikipedia and SUMO knowledge graphs

of 0.89 from SUMO and 0.94 from Wikipedia, leading us to conclude that the Wikipedia KG again resulted in similar performance to the SUMO KG.

In the fourth and fifth experiments, we considered the case of warehouse vs. workroom. The ADE20K training set has 55 warehouse images and four workroom images. To produce a generalized rule to explain warehouse images SUMO returned average coverage of 0.24, while Wikipedia returned 0.56, a significantly larger difference than in previous cases. A large number of positive images compared to that of negative images (55 to 4) may explain the improved coverage score for the Wikipedia KG, as its depth and breadth of concepts exceeds those of SUMO. In the converse experiment (experiment 5)—describing the workroom scenes compared to the warehouse scenes—Wikipedia returned an average coverage score of 0.93 and SUMO returned 0.84. In this case, only four images were used to describe the workroom class, with 55 images on the negative side. Here Wikipedia and SUMO produced comparable average coverage scores.

The results are visualized in Figure 5, showing the simple Wikipedia category hierarchy's superior performance in all experiments compared to the SUMO ontology.

4.2 Noncyclic WKG Information Loss

For the second type of experiment, we evaluated the noncyclic WKG class hierarchy with respect to the DBpedia schema and SUMO knowledge graph to see what proportion of subclass-superclass axioms remain in the WKG compared to the SUMO and DBpedia after breaking cycles. We expected that some subclass-superclass relations would be lost in the cycle-breaking process and hence not exist in our noncyclic WKG despite being present in other KGs. However, our experimental results show little to no information loss, with a substantial majority of the subclass-superclass relations in SUMO and DBpedia preserved in the noncyclic WKG.

The experiment involved first finding matching concepts in the WKG, SUMO, and DBpedia schema. To match the concepts we used a string similarity measurement algorithm (specifically Levenshtein [13] distance=0), finding 22 matching concepts, shown in Table 3. We extracted the asserted superclasses of those concepts from all three KGs. Details of the parents are shown on table 3. In the WKG, the number of asserted parents for some categories are quite large. For example, the category *Fish* has 114 asserted parent categories in the noncyclic WKG. As such, here we show only some of the parent concepts for each category.[10]

Table 3: Parents of all matching concepts in SUMO, DBpedia and noncyclic Wikipedia knowledge graph

Concept	Parent concepts			#Wikipedia parent concepts
	SUMO	DBpedia	Wikipedia	
Aircraft	Vehicle	MeanOfTransportation	Vehicles_by_type, Technology	5
Beer	AlcoholicBeverage	Beverage, Food	Food_and_drink	5
Birth	OrganismProcess	PersonalEvent, LifeCycleEvent, Event	Life	3
Boxing	Sport, ViolentContest	Sport, Activity	Sports	5
Brain	AnimalAnatomicalStructure, Organ	AnatomicalStructure	Human_anatomy, Physical_objects	15
Building	StationaryArtifact	ArchitecturalStructure, Place	Construction, Engineering	12
Cheese	PreparedFood, DairyProduct	Food	Foods	7
City	LandArea, GeopoliticalArea	Settlement, PopulatedPlace, Place	Human_habitats	42
Currency	FinancialInstrument	Thing	International_trade	60
Death	OrganismProcess	PersonalEvent, LifeCycleEvent, Event	Life	3
Fish	ColdBloodedVertebrate	Animal, Eukaryote, Species	Aquatic_organisms	114
Grape	Fruit	FloweringPlant, Plant, Eukaryote, Species	Edible_fruits	20
Language	LinguisticExpression	Thing	Culture	3
Medicine	BiologicallyActiveSubstance	Thing	Health_care, Health	4
Opera	DramaticPlay	MusicalWork, Work	Performing_arts, Entertainment	7
Painting	Coloring, Covering	Artwork, Work	Arts	7
Sales	Working	Activity	Marketing, Business	5
Sculpture	ArtWork	Artwork, Work	Visual_arts, Culture	7
Sound	BodyOfWater	Document, Work	Consciousness, Mind	5
Spacecraft	Vehicle	MeanOfTransportation	Spaceflight	13
Tax	ChargingAFee	TopicalConcept	Governmet_finances	4
Wine	AlcoholicBeverage, PlantAgriculturalProduct	Beverage, Food	Fermented_drinks	32

Due to space constraints, we discuss only a subset of the 22 concepts that matched across the three KGs. We can divide the 22 concepts into twelve subsets by using the first letter of those concepts; among these, the letter *B* has the largest subset, with five elements: *Beer*, *Birth*, *Boxing*, *Brain*, and *Building*.

The concept *Beer* is available in SUMO, DBpedia and WKG. The only SUMO axiom related to the concept *Beer* is $Beer \sqsubseteq AlcoholicBeverage$, while in DBpedia we have $Beer \sqsubseteq Beverage$ and $Beer \sqsubseteq Food$; finally, in the noncyclic WKG we have the related axioms $Beer \sqsubseteq Food\_and\_drink$. We see that all three KGs have semantically similar parents of varying specificity.

[10] See `https://github.com/md-k-sarker/Wiki-KG` for full results.

Axioms related to the concept *Birth* in DBpedia are $Birth \sqsubseteq LifeCycleEvent$, $Birth \sqsubseteq PersonalEvent$ and $Birth \sqsubseteq Event$; in SUMO we have $Birth \sqsubseteq OrganismProcess$; and in the WKG, $Birth \sqsubseteq Life$. We can see that these parent concepts are again similar in meaning.

In SUMO, axioms related to the concept *Boxing* are $Boxing \sqsubseteq Sport$ and $Boxing \sqsubseteq ViolentContest$; DBpedia has $Boxing \sqsubseteq Sport$ and $Boxing \sqsubseteq Activity$; WKG has $Boxing \sqsubseteq Sports$, among others. The parent concepts of *Boxing* are *Sport*, *Sport*, and *Sports* in SUMO, DBpedia, and WKG, respectively; all of these clearly have the same meaning. Minor changes like the pluralization of the category name in Wikipedia are to be expected, as the SUMO and DBpedia schema are manually curated by domain experts and ontologists, while Wikipedia categories are editable by the general public.

Brain is another concept common to all three KGs. In SUMO we have $Brain \sqsubseteq AnimalAnatomicalStructure$ and $Brain \sqsubseteq Organ$, and in DBpedia, $Brain \sqsubseteq AnatomicalStructure$. Some related axioms in WKG are $Brain \sqsubseteq Human\_anatomy$ and $Brain \sqsubseteq Physical\_objects$. We see that ontologically, there exist some differences between *Human_anatomy* and *AnatomicalStructure*, but similar differences also exist between SUMO and DBpedia.

Finally, axioms related to the *Building* concept are: in SUMO, $Building \sqsubseteq StationaryArtifact$; in DBpedia, $Building \sqsubseteq ArchitecturalStructure$ and $Building \sqsubseteq Place$; and in WKG, ten axioms dealing with direct parents of the concept, including $Building \sqsubseteq Construction$ and $Building \sqsubseteq Society$. We again see that the parents are similar in semantics, though slight differences exist among the three ontologies.

Based on the above, we conclude that there is minimal information loss in the noncyclic Wikipedia KG with respect to DBpedia and SUMO. There exist some minor differences in an ontological sense with the WKG axioms, but such minor differences exist between SUMO and DBpedia as well.

5 Conclusion

The readily available Wikipedia category hierarchy and its corresponding named entities has great importance in artificial intelligence and its subfields. We make the Wikipedia Knowledge Graph (WKG), break its cycles, and make available both the original and cycle-free versions for public use. We evaluate the WKG in the context of XAI and compare it with the DBpedia and SUMO KGs, finding WKG to be highly effective compared to the other two. We also evalute the noncyclic WKG relative to SUMO and the DBpedia schema, finding minimal information loss. Here we evaluate the WKG in a specific XAI application; further work should focus on evaluating it in other such applications and in different domains of artificial intelligence.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia—A crystallization point for the Web of Data. Journal of Web

Semantics **7**(3), 154–165 (2009)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: In SIGMOD Conference. pp. 1247–1250 (2008)
3. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner – A framework for inductive learning on the semantic web. J. Web Sem. **39**, 15–24 (2016)
4. Confalonieri, R., del Prado, F.M., Agramunt, S., Malagarriga, D., Faggion, D., Weyde, T., Besold, T.R.: An ontology-based approach to explaining artificial neural networks (2019)
5. Doran, D., Schulz, S., Besold, T.R.: What does explainable AI really mean? A new conceptualization of perspectives. In: Besold, T.R., Kutz, O. (eds.) Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017, Bari, Italy, 2017. CEUR Workshop Proceedings, vol. 2071. CEUR-WS.org (2017)
6. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
7. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 93 (2018)
8. Gunning, D.: Explainable artificial intelligence (XAI). Defense Advanced Research Projects Agency (DARPA) (2017)
9. Hitzler, P., Bianchi, F., Ebrahimi, M., Sarker, M.K.: Neural-symbolic integration and the semantic web. Semantic Web (2020), accepted for publication
10. Holzinger, A., Biemann, C., Pattichis, C.S., Kell, D.B.: What do we need to build explainable AI systems for the medical domain? (2017)
11. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic web **2**(1), 11–21 (2011)
12. Lecue, F.: On the role of knowledge graphs in explainable AI. Semantic Web journal (2019), http://www.semantic-web-journal.net/system/files/swj2198.pdf, retrieved on July 26, 2019
13. Levenshtein, V.I.: On the minimal redundancy of binary error-correcting codes. Inf. Control. **28**(4), 268–291 (1975). https://doi.org/10.1016/S0019-9958(75)90300-9, `https://doi.org/10.1016/S0019-9958(75)90300-9`
14. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. The Journal of Logic Programming **19-20**, 629 – 679 (1994). https://doi.org/https://doi.org/10.1016/0743-1066(94)90035-3, `http://www.sciencedirect.com/science/article/pii/0743106694900353`, special Issue: Ten Years of Logic Programming
15. Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001. pp. 2–9 (2001)
16. Samek, W., Wiegand, T., Müller, K.: Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. CoRR **abs/1708.08296** (2017), `http://arxiv.org/abs/1708.08296`
17. Sarker, M.K., Hitzler, P.: Efficient concept induction for description logics. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3036–3043 (2019)
18. Sarker, M.K., Xie, N., Doran, D., Raymer, M., Hitzler, P.: Explaining trained neural networks with semantic web technologies: First steps. In: Besold, T.R., d'Avila Garcez, A.S., Noble, I. (eds.) Proceedings of the Twelfth International

Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017. CEUR Workshop Proceedings, vol. 2003. CEUR-WS.org (2017)
19. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. ACM Press, New York, NY, USA (2007)
20. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 5122–5130. IEEE Computer Society (2017)

A Steps for Building the Wikipedia Knowledge Graph

As of 20 January 2020, the *page* table[11] (containing article information) has around 49 million entries, while the *categorylinks* table[12] (containing category information) has around 140 million entries.

As these files are large (the larger is 24GB), proper settings must be applied to the database before importing them to keep the import process from taking a prohibitively long time. In particular, we must disable foreign key checking and increase the buffer length.

There are different types of pages on Wikipedia: some pages are articles, some pages are categories, and some pages are for administrative use. Administrative pages are not of interest for the knowledge graph, so we omit them. Using the information from the table *categorylinks*, we can identify which pages are articles, which are categories, and so on. The column *page_namespace* holds the page type information; for categories, *page_namespace=14*, while for articles, *page_namespace=0*. This table also provides the category hierarchical information, in its columns *cl_from* and *cl_to*. The column *cl_from* is the article name or subcategory name, and column *cl_to* is the category or parent category name (depending on whether the page is an article or category). Each page has a unique ID and title. The table *page* gives us the needed information like ID of the page, title, etc.

The steps to create the knowledge graph are shown in Algorithm 1. By way of example, we demonstrate part of the execution of Algorithm 1 on the article Albert_Einstein.[13] Initially, we need to get the page_id for Albert_Einstein from the *page* table downloaded from the dump by executing the following query.

```
SELECT page_id, page_title, page_namespace FROM page
WHERE page_title = `Albert_Einstein' and page_namespace = 0;
```

[11] Available for download at `http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-page.sql.gz`, with and described in detail at `https://www.mediawiki.org/wiki/Manual:Page_table`.

[12] Available for download at `http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-categorylinks.sql.gz`, and described in detail at `https://www.mediawiki.org/wiki/Manual:Categorylinks_table`.

[13] `https://en.wikipedia.org/wiki/Albert_Einstein`

Algorithm 1: Wikipedia knowledge graph construction algorithm

```
1  Function Iterate(A) :
2      Find page_id pd, title t, page_namespace pn of page A;
3      if pn == 0 then
4          Declare title t as an entity e;
5          Find categories ($c \in C$) of entity e;
6          foreach $c \in C$ do
7              Declare category c as a rdf:type (class);
8              Create facts: e rdf:type c;
9              Find the pages ($p \in P$) which are entity of category c;
10             foreach $p \in P$ do
11                 Iterate(p) ;
12             end
13         end
14     end
15     else if pn == 14 then
16         Declare title t a category (class) c;
17         Find all sub-categories ($sc \sqsubseteq c$) of category c;
18         foreach $sc \in C$ do
19             Create relation: $sc$ subClassOf c;
20             Iterate($sc$);
21         end
22     end
23 end
24 Iterate(Main_topic_classifications)      /* start the process from root */
```

The result of this query is in figure 6, and we can see that the page_id of article Albert_Einstein is 736.

After getting the page_id, we need to get the page's category, which we can get using the following query.

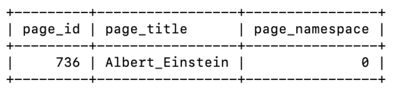
```
SELECT cl_from, cl_to FROM categorylinks WHERE cl_from = 736;
```

As of 20 January 2020, this page belongs to 148 different categories, a subset of which is shown in Figure 7.

Using the results of these queries, we can create axioms like *Albert_Einstein rdf:type German_inventors* and incorporate them into our knowledge graph. To continue creating the full hierarchy, we must continue with the parent categories of each the article's categories.

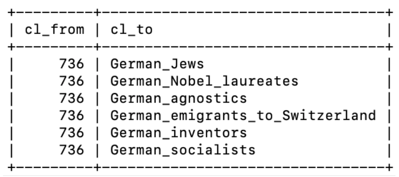
To get the parent category of a category, we must find the page_id of that category and use that to find its parent. For example, if we want to find the parent category of German_inventors, we need to determine the page_id of the German_inventors page as follows.

```
SELECT page_id, page_title, page_namespace FROM page
WHERE page_title = `German_inventors' and page_namespace = 14;
```



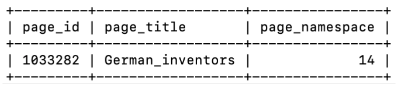
```
+---------+-----------------+----------------+
| page_id | page_title      | page_namespace |
+---------+-----------------+----------------+
|     736 | Albert_Einstein |              0 |
+---------+-----------------+----------------+
```

Fig. 6: Page_id of the article Albert_Einstein



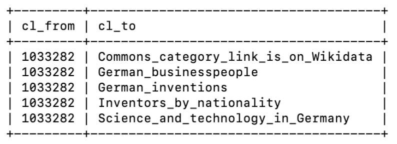
```
+---------+---------------------------------+
| cl_from | cl_to                           |
+---------+---------------------------------+
|     736 | German_Jews                     |
|     736 | German_Nobel_laureates          |
|     736 | German_agnostics                |
|     736 | German_emigrants_to_Switzerland |
|     736 | German_inventors                |
|     736 | German_socialists               |
+---------+---------------------------------+
```

Fig. 7: Categories for the article Albert_Einstein



```
+---------+------------------+----------------+
| page_id | page_title       | page_namespace |
+---------+------------------+----------------+
| 1033282 | German_inventors |             14 |
+---------+------------------+----------------+
```

Fig. 8: Page_id of category *German_inventors*



```
+---------+-------------------------------------+
| cl_from | cl_to                               |
+---------+-------------------------------------+
| 1033282 | Commons_category_link_is_on_Wikidata |
| 1033282 | German_businesspeople               |
| 1033282 | German_inventions                   |
| 1033282 | Inventors_by_nationality            |
| 1033282 | Science_and_technology_in_Germany   |
+---------+-------------------------------------+
```

Fig. 9: Parent categories of the category German_inventors

This will return the result shown in Figure 8, where we see that the page_id of German_inventors is 1033282.

After getting this page_id, we can consult the *categorylinks* table for the parent category:

```
SELECT cl_from, cl_to FROM categorylinks WHERE cl_from = 1033282;
```

This will provide the parent results as shown in Figure 9, where we see that the parent categories of *German_inventors* are *Inventors_by_nationality* and *Science_and_technology_in_Germany*, among others.[14] This kind of relationship creates cycles in the category hierarchy, as discussed in Section 3.2.

We now see the complete process of creating an entity and adding axioms for its types and supertypes. The example above is but one fragment of the knowledge graph creation adventure; to complete the knowledge graph, we need to start from the root of the category hierarchy and continue with Algorithm 1 until all pages have been processed to yield article titles with their categories, along with the resulting category hierarchy.

[14] It may seem odd to have *Science_and_technology_in_Germany* and similar as parent categories of *German_inventors* in an ontology; this reflects the somewhat messy nature of Wikipedia.