Studies on the Semantic Web

Ilaria Tiddi, Freddy Lécué
and Pascal Hitzler (Eds.)

Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges

IOS Press

AKA

ISBN 978-3-89838-754-5 (AKA, print)
ISBN 978-1-64368-080-4 (IOS Press, print)
ISBN 978-1-64368-081-1 (IOS Press, online)
doi: 10.3233/SSW47

Bibliographic information available from the Katalog der Deutschen Nationalbibliothek (German National Library Catalogue) at https://www.dnb.de

Publisher
Akademische Verlagsgesellschaft AKA GmbH, Berlin

Represented by Co-Publisher IOS Press
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
The Netherlands
Tel: +31 20 688 3355
Fax: +31 20 687 0019
email: order@iospress.nl

Contents

Part 3. Challenges for Knowledge-Based eXplainable Systems

Part 1

Foundations of Knowledge-Based eXplainable Systems

This page intentionally left blank

Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges 3
I. Tiddi et al. (Eds.)
IOS Press, 2020

doi:10.3233/SSW200009

Knowledge Graphs on the Web – An Overview

Nicolas HEIST, Sven HERTLING, Daniel RINGLER, and Heiko PAULHEIM
Data and Web Science Group, University of Mannheim, Germany

Abstract. Knowledge Graphs are an emerging form of knowledge representation. While Google coined the term *Knowledge Graph* first and promoted it as a means to improve their search results, they are used in many applications today. In a knowledge graph, entities in the real world and/or a business domain (e.g., people, places, or events) are represented as nodes, which are connected by edges representing the relations between those entities. While companies such as Google, Microsoft, and Facebook have their own, non-public knowledge graphs, there is also a larger body of publicly available knowledge graphs, such as DBpedia or Wikidata. In this chapter, we provide an overview and comparison of those publicly available knowledge graphs, and given insights into their contents, size, coverage, and overlap.

Keywords. Knowledge Graphs, Linked Data, Semantic Web, Dataset Profiling

1. Introduction

Knowledge Graphs are increasingly used as means to represent knowledge. Due to their versatile means of representation, they can be used to integrate different heterogeneous data sources, both within as well as across organizations [8,9].

Besides such domain-specific knowledge graphs which are typically developed for specific domains and/or use cases, there are also public, cross-domain knowledge graphs encoding common knowledge, such as DBpedia, Wikidata, or YAGO [33]. Such knowledge graphs may be used, e.g., for automatically enriching data with background knowledge to be used in knowledge-intensive downstream applications [34]. In particular for the case of eXplainable AI, knowledge graphs can be used as additional input to the AI algorithm, as a means to support interpretation of the results, or both [18].

Since Google coined the term *Knowledge Graph* for marketing purposes, it has subsequently been used in the scientific literature as well. The slogan by which Google announced KGs was *Things, not Strings*[1]. The idea of that slogan is: while strings are often ambiguous, knowledge graphs consist of disambiguated entities, so that entities of the same name can be told apart more easily. Nowadays, almost all companies processing large amounts of heterogeneous data use knowledge graphs as a means of representation, including, but not limited to IBM, Microsoft, Facebook or Ebay [27].

There are quite a few different definitions for knowledge graphs [5]. Typically, a knowledge graph

[1] https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

1. mainly describes real world entities and their interrelations, organized in a graph.
2. defines possible classes and relations of entities in a schema.
3. allows for potentially interrelating arbitrary entities with each other.
4. covers various topical domains [29].

In this chapter, we provide an overview of publicly available, cross-domain knowledge graphs on the Web. We discuss the techniques used to create those knowledge graphs and provide an in-depth comparison in terms of size, level of detail, contents, and overlap.

2. Overview

There are different techniques for creating knowledge graphs. The most common ones are (1) manual curation, (2) creation from (semi) structured sources, and (3) creation from unstructured sources. Some knowledge graphs also use a mix of those techniques.

2.1. Manual Curation

Cyc [20] is one of the oldest knowledge graphs; the Cyc project dates back to the 1990s. Cyc was created along with its own language (CycL), which provides a large degree of formalization.

While Cyc was developed by a comparatively small group of experts, the idea of *Freebase* [32] was to establish a large community of volunteers, compared to Wikipedia. To that end, the schema of Freebase was kept fairly simple to lower the entrance barrier as much as possible. Freebase was acquired by Google in 2010 and shut down in 2014.

Wikidata [43] also uses on a crowd editing approach. In contrast to Cyc and Freebase, Wikidata also imports entire whole large datasets, such as several national libraries' bibliographies. Porting the data from Freebase to Wikidata is also a long standing goal [32].

Curating a knowledge graph manually can be a large effort. The total cost of development for Cyc have been estimated as 120 Million USD[2]. This corresponds to a total cost of 2-6 USD per single axiom in Cyc [30].

2.2. Creation from (Semi) Structured Sources

A more efficient way of knowledge graph creation is the use of structured or semi structured sources. Wikipedia is a commonly used starting point for knowledge graphs such as *DBpedia* [19] and *YAGO* [40].

DBpedia mainly uses infoboxes in Wikipedia. Those are manually mapped to a predefined ontology; the mapping is crowd sourced using a Wiki and a community of volunteers. Given those mappings, the DBpedia Extraction Framework creates a graph in which each page in Wikipedia becomes an entity, and all values and links in an infobox become attributes and edges in the graph.

YAGO uses a similar process, but classifies instances based on the category structure and WordNet [24] instead of infoboxes. YAGO integrates various language editions of

[2] http://www.ttivanguard.com/conference/Napa2017/4-Lenat.pdf

Wikipedia into a single graph and represents temporal facts with meta-level statements, i.e., RDF reification.

CaLiGraph also uses information in categories, but aims at converting them into formal axioms using DBpedia as supervision [11]. Moreover, instances from Wikipedia list pages are considered for populating the knowledge graph [31]. The result is a knowledge graph which is not only richly populated on the instance level, but also has a large number of defining axioms for classes [12].

A similar approach, i.e., the combination of information in Wikipedia and WordNet, is used by *BabelNet* [25]. The main purpose of BabelNet is the collection of synonyms and translations in various languages, so that this knowledge graph is particularly well suited for supporting multi-language applications. Similarly, *ConceptNet* [38] collects synonyms and translations in various languages, integrating multiple third party knowledge graphs itself.

DBkWik [14] uses the same codebase as DBpedia, but applies it to a multitude of Wikis. This leads to a graph which has a larger coverage and level of detail for many long tail entities, and is highly complementary to DBpedia. However, the absence of a central ontology and mappings, as well as the existence of duplicates across Wikis, which might not be trivial to detect, imposes a number of challenges not present in DBpedia.

Another source of structured data is the structured annotations in Web pages using techniques such as RDFa, Microdata, and Microformats [23]. While the pure collection of those could, in theory, already be considered a knowledge graph, that graph would be rather disconnected and consist of a plethora of small, unconnected components [28] and would require additional cleanup for compensating irregular use of the underlying schemas and shortcomings in the extraction [22]. A consolidated version of this data into a more connected knowledge graph has been published under the name *VoldemortKG* [42].

2.3. Creation from Unstructured Sources

The extraction of a knowledge graph from semi structured sources is considered more easy than from the extraction from unstructured sources. However, there is much more information in unstructured sources (such as text). Therefore, extracting knowledge from unstructured sources has also been proposed.

NELL [4] is an example for extracting a knowledge graph from free text. NELL was originally trained with a few seed examples and continuously runs an iterative coupled learning process. In each iteration, facts are used to learn textual patterns to detect those facts, and patterns learned in previous iterations are used to extract new facts, which serve as training examples in later iterations. To improve the quality, NELL has introduced a feedback loop incorporating occasional human feedback.

WebIsA [37] also extracts facts from free text, but focuses on the creation of a large-scale taxonomy. For each extracted fact, rich metadata are collected, including the sources, the original sentences, and the patterns used in the extraction of a particular fact. Those metadata are exploited for computing a confidence score for each fact [13].

3. Comparison of Knowledge Graphs

Whenever a knowledge graph is to be used in an application, it is important to determine which knowledge graph is best suitable for an application at hand. The knowledge graphs mentioned above differ in their content, their level of detail, etc. Hence, in this chapter, we will discuss several characteristics of knowledge graphs and provide insights into the differences between them.

3.1. General Metrics

The most straightforward metrics to be used consider the mere amount of information contained in a knowledge graph. Measures that may be used include:

- The number of instances in a graph
- The number of assertions (or edges between entities)
- The average and median linkage degree (i.e.: how many assertions per entity does the graph contain?)

As for using a knowledge graph in an XAI system, these metrics hint at the utility – the more information about the domain at hand is present (i.e., the more instances are represented in the knowledge graph and the more detailed that information is), the more can an XAI application benefit in providing better results or better interpretations.

Another set of metrics can be defined for the schema or ontology level of a knowledge graph:

- The number of classes defined in the schema
- The number of relations defined in the schema
- The average depth and width (branching factor) of the class hierarchy[3]
- The complexity of the schema

While the instance-based metrics focus more on the coverage of a domain in a knowledge graph, these schema-level metrics provide information about the richness and formality of that knowledge. They determine which techniques to use – e.g., while more formal, very complex ontologies will call for using ontology reasoning, light-weight, but large-scale ontologies will be better exploited by statistical and distributional approaches.

Table 1 depicts those metrics for some of the knowledge graphs discussed above. ConceptNet and WebIsA are not included, since they do not distinguish a schema and instance level (i.e., there is no specific distinction between a class and an instance), which does not allow for computing those metrics meaningfully. For Cyc, which is only available as a commercial product today, we used the free version OpenCyc, which has been available until 2017.[4]

From those metrics, it can be observed that the KGs differ in size by several orders of magnitude. The sizes range from 50,000 instances (and Voldemort) to 50 million instances (for Wikidata), so the latter is larger by a factor of 1,000. The same holds for assertions. Concerning the linkage degree, YAGO is much richer linked than the other graphs.

[3]While this could also be done for the property hierarchy, extensive property hierarchies are rather rare in common knowledge graphs.

[4]It is still available, e.g., at https://github.com/asanchez75/opencyc

Table 1. Basic Metrics of Open Knowledge Graphs

	DBpedia	YAGO	Wikidata	BabelNet
# Instances	5,044,223	6,349,359	52,252,549	7,735,436
# Assertions	854,294,312	479,392,870	732,420,508	178,982,397
Avg. linking degree	21.30	48.26	6.38	0.00
Median ingoing edges	0	0	0	0
Median outgoing edges	30	95	10	9
# Classes	760	819,292	2,356,259	6,044,564
# Relations	1355	77	6,236	22
Avg. depth of class tree	3.51	6.61	6.43	4.11
Avg. branching factor of class tree	4.53	8.48	36.48	71.0
Ontology complexity	SHOFD	SHOIF	SOD	SO
	Cyc	NELL	CaLiGraph	Voldemort
# Instances	122,441	5,120,688	7,315,918	55,861
# Assertions	2,229,266	60,594,443	517,099,124	693,428
Avg. linking degree	3.34	6.72	1.48	0
Median ingoing edges	0	0	0	0
Median outgoing edges	3	0	1	5
# Classes	116,821	1,187	755,963	621
# Relations	148	440	271	294
Avg. depth of class tree	5.58	3.13	4.74	3.17
Avg. branching factor of class tree	5.62	6.37	4.81	5.40
Ontology complexity	SHOIFD	SROIF	SHOD	SH

Figure 1 shows an overview of the knowledge graphs considered. We follow the conventions of the Linked Open Data Cloud diagrams[5] [36], which are used to depict linked datasets and their connections. In those diagrams, the size of the circles is proportional to the number of instances, and the strength of the connecting lines is proportional to the number of links.

The knowledge graphs also differ strongly in the characteristics of their schema. DBpedia and NELL have comparably small schemas, while Wikidata and BabelNet build deep and detailed taxonomies. For example, while NELL does not define detailed subclasses for *Scientist*[6], DBpedia defines four subclasses[7], Wikidata has more than 600[8] and CaLiGraph almost 2,000[9], including detailed classes such as *sickle-cell disease researcher* or *loop quantum gravity researcher*. Voldemort, on the other hand, reuses the schema.org ontology, which is comparably small [21].

Looking at the complexity, it is not much of a surprise that Cyc, originating in classic AI research and strongly building on logical rules [3,20], has the highest complexity. Wikidata, BabelNet, and Voldemort have only little complexity, the other graphs are somewhere inbetween.

[5] https://www.lod-cloud.net/
[6] http://rtw.ml.cmu.edu/rtw/kbbrowser/pred:scientist
[7] http://dbpedia.org/ontology/Scientist
[8] https://www.wikidata.org/wiki/Q15976092
[9] http://caligraph.org/ontology/Scientist

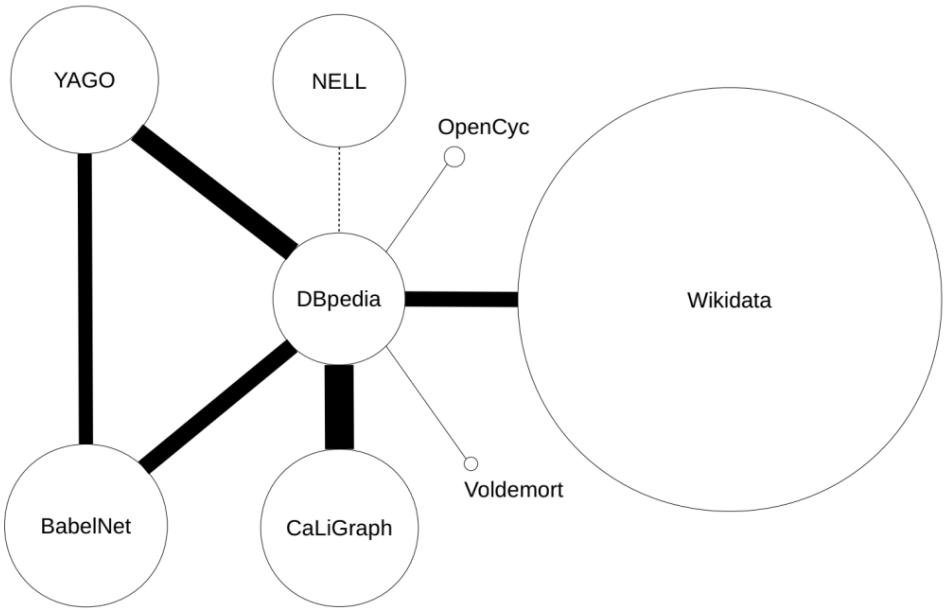

Figure 1. Depiction of the size and linkage degree of publicly available knowledge graphs. Although NELL and DBpedia are not explicitly interlinked, NELL contains links to Wikipedia, which can be trivially translated to DBpedia links.

3.2. Contents

The knowledge graphs do not only differ in their size and level of detail, but also in their contents. The most straightforward way to assess the content focus of a knowledge graph is to look at the size of its classes. Figures 2-9 show graphic depictions of those class sizes. The diagrams were created starting from the most abstract class and following the class hierarchy to the largest respective subclasses.

At first glance, the figures reveal differences in the development of the taxonomies. While Cyc builds a formal ontology with very abstract top level categories such as *partially intangible thing* or *thing that exists in time*, the more pragmatic classification in DBpedia and Voldemort (the latter using schema.org as an ontology) has top level classes such as *Place* or *Person*. The reason for these differences lies in the origins of the respective knowledge graphs: While Cyc's classification was created by AI researchers, the ontology in DBpedia is the result of a crowdsourcing process [30]. The same holds for schema.org, which is a pragmatic effort of a consortium of search engine developers.

Moreover, the diagrams reveal some differences in the contents. The main focus of DBpedia is on persons (and their careers), as well as places, works, and species. Wikidata also has a strong focus on works (mainly due to the import of entire bibliographic datasets), while Cyc, BabelNet and NELL show a more diverse distribution.

3.3. Looking into Details

To obtain deeper insights which classes are more prominent in which KGs, and, ultimately, which KGs are suitable for building eXplainable AI system in a specific domain,

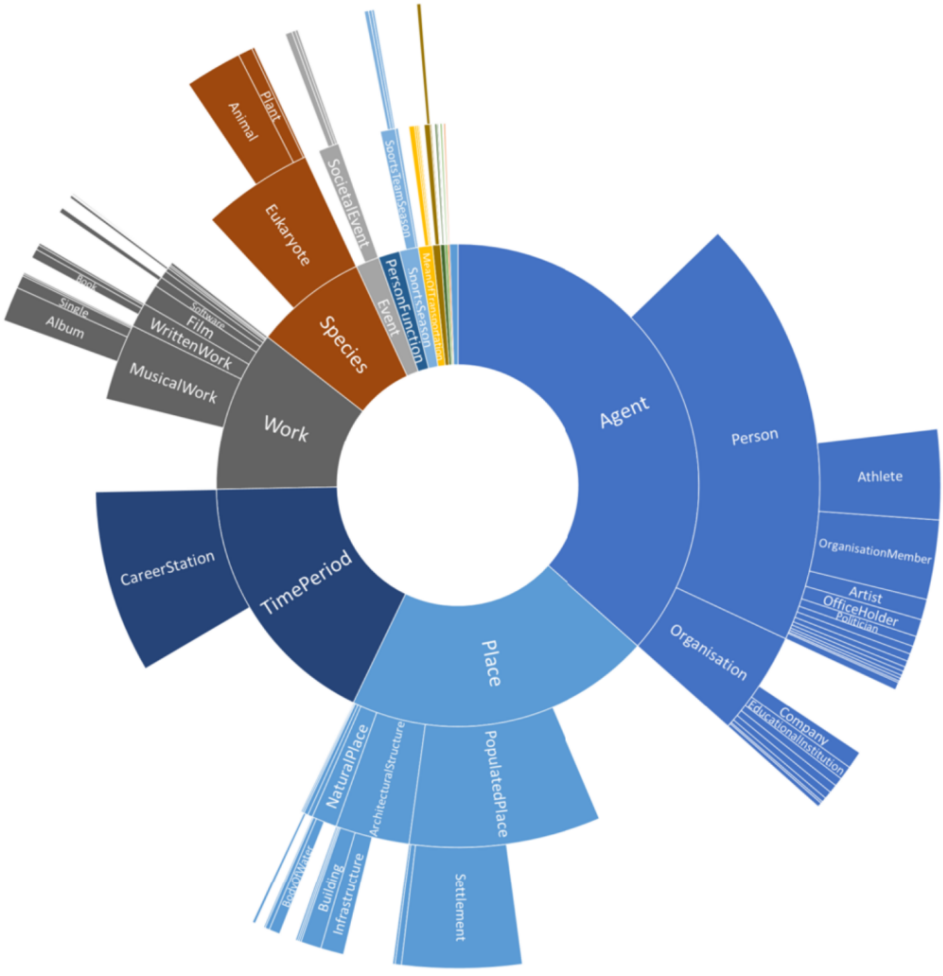

Figure 2. Instances in DBpedia

it is useful to not only look at the number of instances, but also the level of detail in which those instances are represented (i.e., the linkage degree and number of assertions per instance).

Table 2 depicts such a detailed view for ten prominent classes:

- Person
- Organization
- Populated place (city, country, etc.)
- Uninhabited place (mountain, lake, etc.)
- Species
- Work (book, movie, etc.)
- Building
- Gene
- Protein
- Event

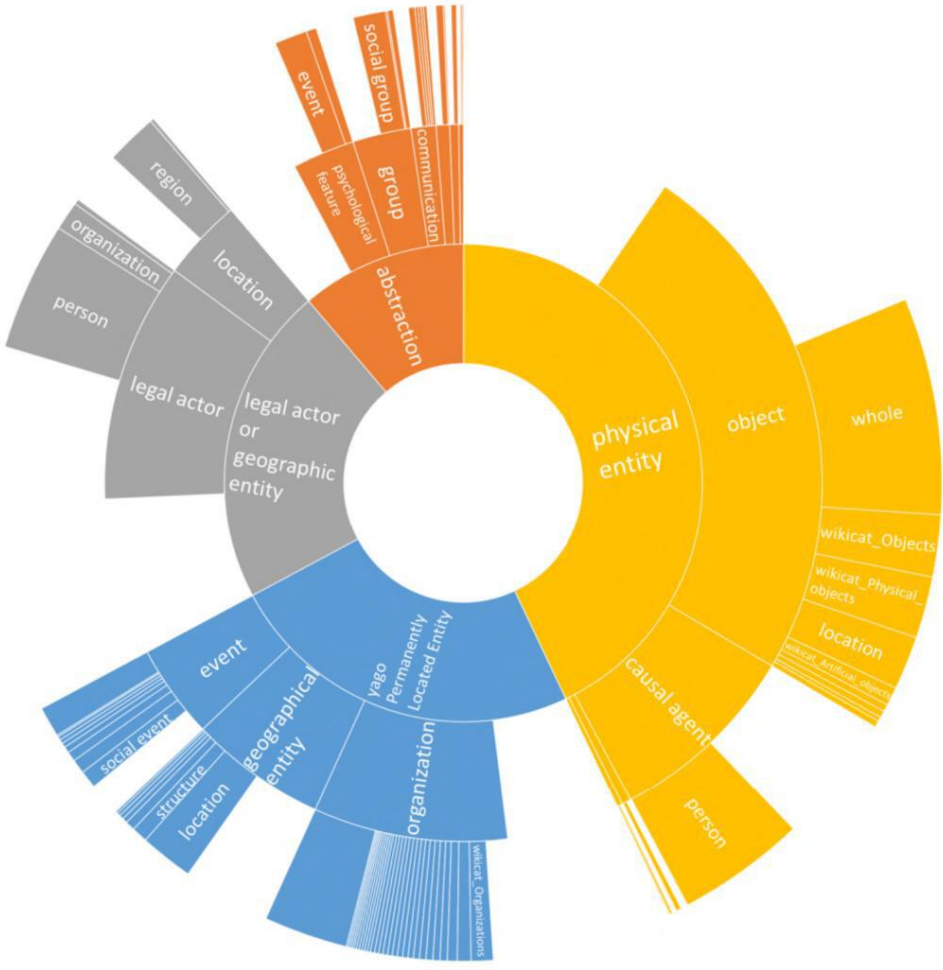

Figure 3. Instances in YAGO

The global trend observed in this table is that Wikidata has the largest number of instances in most of the classes, while YAGO has the largest level of detail. However, there are differences from class to class. While Wikidata has a large number of works, YAGO is a good source of events. NELL often has fewer instances, but a larger level of detail, which can be explained by its focus on more prominent instances.

The contrast of the average and the median degree also reveals a few differences. For example, BabelNet contains a similar amount of instances as DBpedia for some classes, e.g., uninhabited places or works. While the average linkage degree is higher in DBpedia, the median is higher in BabelNet. This hints at a more uneven distribution of information in DBpedia, while BabelNet has a more constant distribution of statements per instance.

Table 2.: Detail statistics for selected classes

	DBpedia				YAGO				Wikidata			
Class	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out
Person	1,243,400	1.54	0	5	2,213,431	5.62	0	258	5,250,840	2.41	0	10
Organization	286,482	10.3	0	7	498,750	136.92	0	64	1,665,319	30.98	0	6
Populated place	513,642	7.38	0	8	319,210	219.49	0	138	2,355,559	3.81	1	6
Uninhabited place	67,495	0.91	0	4	160,615	23.67	0	48	1,516,890	0.23	0	6
Species	306,104	2.56	0	7	2,553,369	4.92	0	224	110	21.53	0	5
Work	496,070	0.81	0	8	1,175,125	28.07	0	36	34,585,828	5.91	0	12
Building	197,831	0.41	0	5	274,606	13.44	0	69	2,291,168	0.98	0	6
Gene	4	0.5	0.5	7.5	12,351	0.00	0	8	172,128	1.27	0	8
Protein	2,747	0.03	0	1	10,935	0.01	0	52	84,163	1.15	1	14
Event	76,029	1.91	0	3	562,583	41.23	0	48	579,559	1.76	0	5

	BabelNet				Cyc				NELL			
Class	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out
Person	2,384,065	0.00	0	17	12,784	0.04	0	3	90,601	8.93	0	0
Organization	764,662	0.01	0	12	26,276	5.70	0	5	41,646	6.31	0	0
Populated place	509,257	0.01	0	9	8,596	20.63	0	12	28,359	39.98	0	0
Uninhabited place	70,209	0.02	0	11	64	2.05	1	12	158,879	3.83	0	0
Species	6,536	0.01	0	17	0	-	-	-	3,273	0.88	0	0
Work	491,057	0.00	0	12	19,908	0.91	0	2	27,038	1.09	0	0
Building	520	0.00	0	8	786	0.14	0	4	50,699	4.51	0	0
Gene	522	0.00	0	5	8	0	0	3	0	-	-	-
Protein	10,399	0.00	0	3	0	-	-	-	0	-	-	-
Event	9,904	0.00	0	13	685	0.86	0	2	37,203	0.65	0	0

	CaLiGraph				Voldemort			
Class	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out
Person	1,967,339	0.34	0	2	36,370	0.00	0	5
Organization	547,728	2.67	0	2	5,984	0.00	0	1
Populated place	700,559	10.12	0	2	1,278	0.00	0	5
Uninhabited place	170,324	1.16	0	2	60	0.00	0	4
Species	552,249	1.04	0	1	0	-	-	-
Work	678,888	0.49	0	1	6,673	0.00	0	3
Building	404,087	0.21	0	1	108	0.00	0	5
Gene	1,106	0.00	0	0	0	-	-	-
Protein	6,138	0.00	0	0	0	-	-	-
Event	148,122	0.49	0	0	198	0.00	0	4

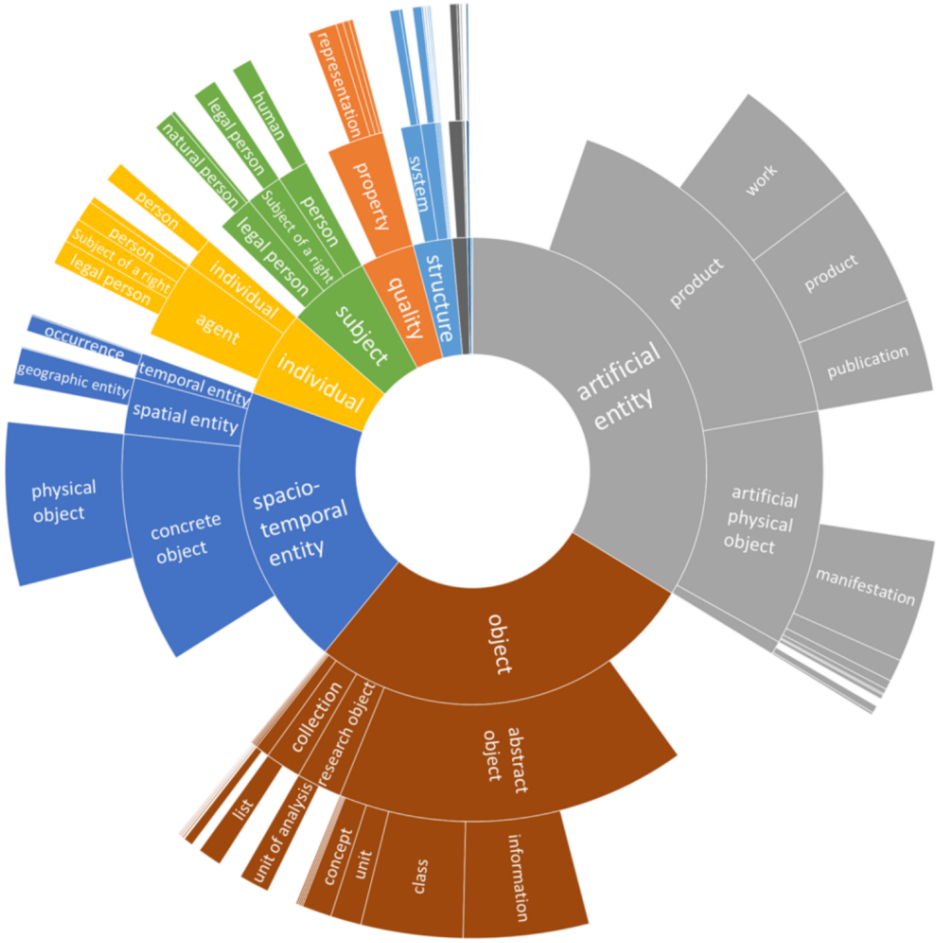

Figure 4. Instances in Wikidata

4. Linkage and Overlap of Knowledge Graphs

Since knowledge graphs differ so strongly in size, coverage, and level of detail, combining information from multiple KGs for implementing one application is often beneficial. To estimate the value of such a combination, we determine the overlap of the knowledge graphs first.

As shown in Fig. 1, many KGs contain explicit interlinks. Those links, usually in the form of `owl:sameAs` links, express that entities in two KGs are the same (or, more precisely: that they refer to the same real world entity) [10]. In other cases, such links can be generated indirectly, e.g., if a knowledge graph contains links to Wikipedia pages, which can be easily mapped to entities in DBpedia and YAGO.

Even if those links provide a first hint at the overlap of KGs, and further links can be found by exploiting the transitivity of the `owl:sameAs` property [1], they do not provide a complete picture. Due to the *open world assumption*, which holds for KG interlinks

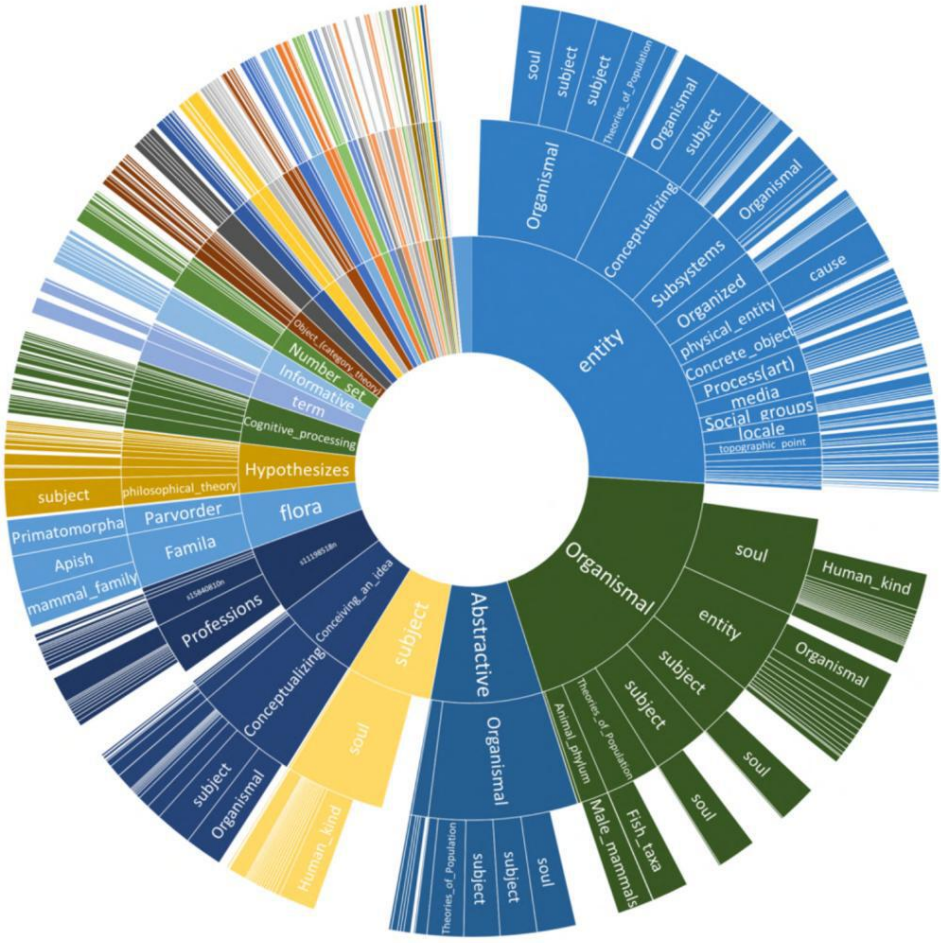

Figure 5. Instances in BabelNet

as well, there might always be more links than the one which are explicitly or implicitly provided by the KGs.

4.1. Method

In order to estimate the actual number of interlinks, we use a method first discussed in [33], which builds on a set of existing links and heuristic link discovery:

1. We use different heuristics to discover links between two KGs automatically, e.g., different string similarity measures [7,26].
2. Based on the existing, incomplete set of interlinks, we measure recall and precision of the individual heuristics [35].
3. With the help of those recall and precision figures, we can estimate the actual number of interlinks. After repeating the procedure with multiple heuristics, we can use the average of those estimations.

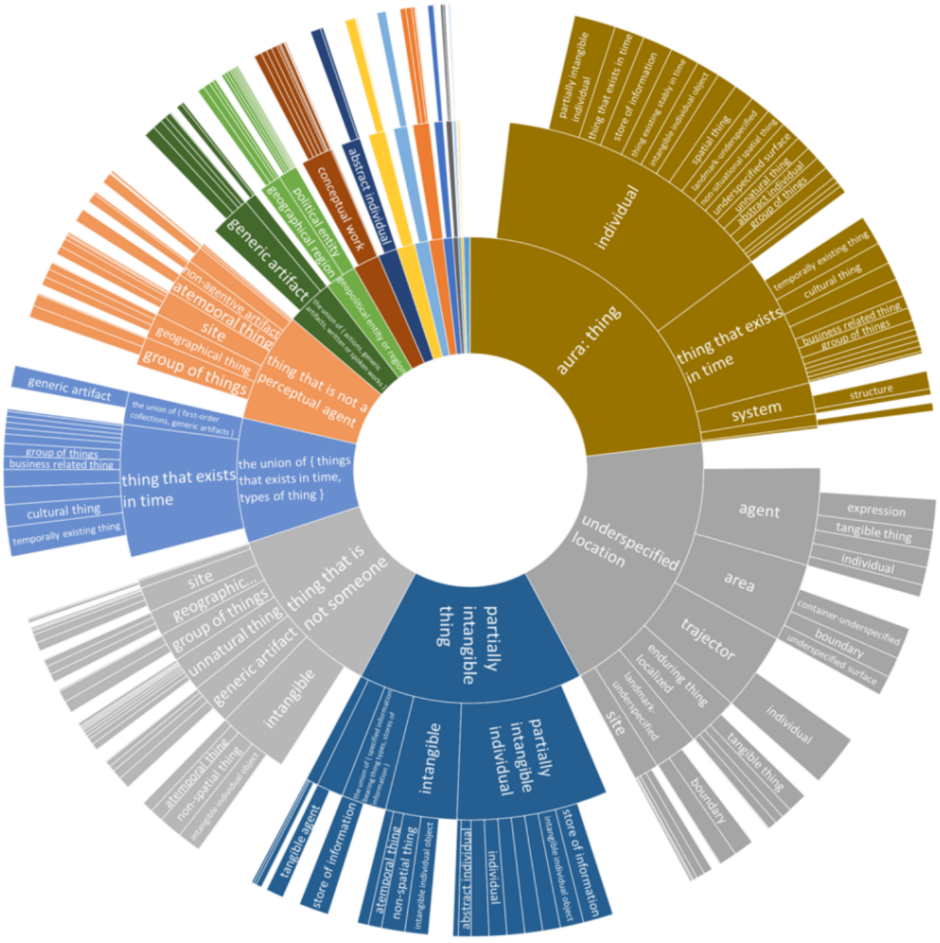

Figure 6. Instances in OpenCyc

Given that the actual number of links is C (which is unknown), the number of links found by a heuristic is F, and that the number of correct links in F is F^+, recall and precision are defined as

$$R := \frac{|F^+|}{|C|} \tag{1}$$

$$P := \frac{|F^+|}{|F|} \tag{2}$$

By resolving both to $|F^+|$ and combining the equations, we can estimate $|C|$ as

$$|C| = |F| \cdot P \cdot \frac{1}{R} \tag{3}$$

Thus, we can obtain an estimate for C given F, R, and P. A more intuitive interpretation of the last equation is that P is a measure of how strongly the heuristic *over*estimates the number of actual interlinks (thus, F is reduced by multiplication with P), and R is

Figure 7. Instances in NELL

a measure of how strongly the heuristic *under*estimates the number of actual interlinks (thus, F is divided by R).

In [33], we have shown that across different heuristics, although F varies a lot, the estimate C is fairly stable. For producing the estimates in this chapter, we have used the following heuristics: string equality, scaled Levenshtein (thresholds 0.8, 0.9, and 1.0), Jaccard (0.6, 0.8, and 1.0), Jaro (0.9, 0.95, and 1.0), JaroWinkler (0.9, 0.95, and 1.0), and MongeElkan (0.9, 0.95, and 1.0). The estimated overlap reported is the average estimate computed using these 16 metrics.

4.2. Findings

To analyze the benefit of the combination of different KGs, we depict the number of estimated links both in relation to (a) the entities existing in the larger of the two KGs (Fig. 10) as well as (b) in relation to the links that exist explicitly or implicitly (Fig. 11). From (a), we can estimate the amount of gain in knowledge of combining two KGs (i.e.,

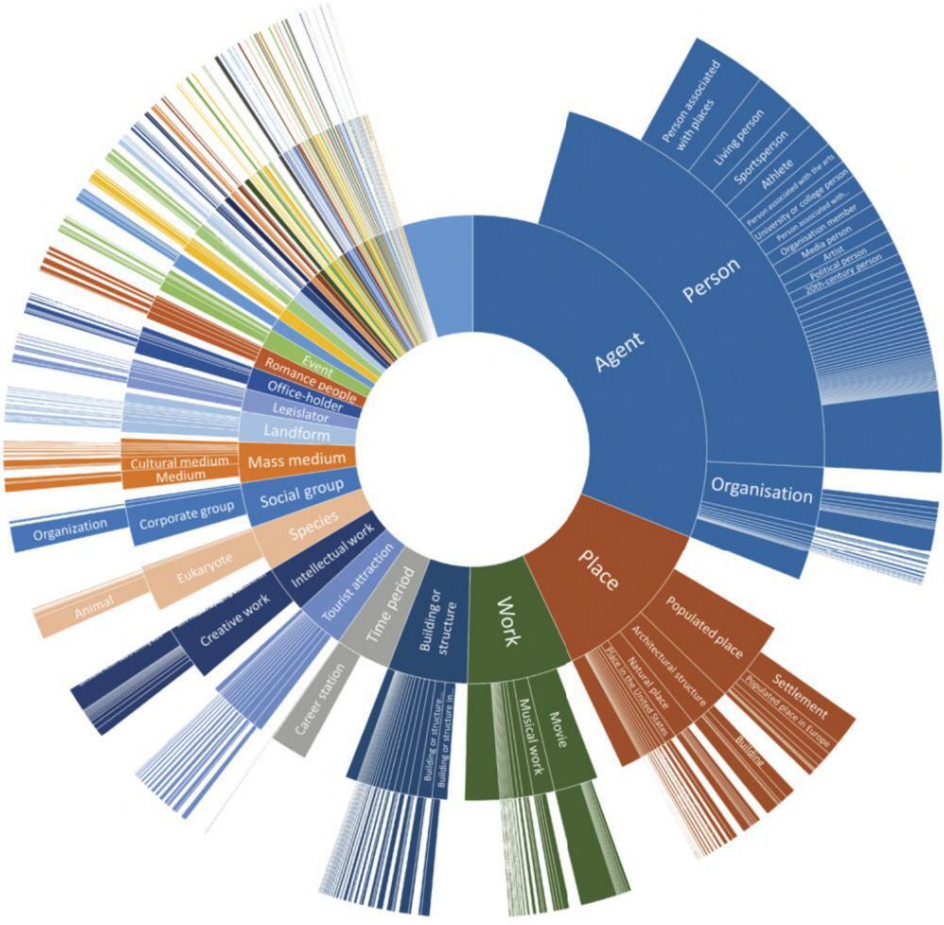

Figure 8. Instances in CaLiGraph

if only a small fraction of one KG is also contained in the other and vice versa, such a combination adds a lot of information). From (b), we can get insights into whether the set of existing links is sufficient for such a combination or not.

Fig. 10 shows that in most cases, the larger of two knowledge graphs contains most of the entities of a smaller one, i.e., its set of entities of a class in larger KG is usually a superset of that set in the smaller one. For example, as depicted in table 2, Wikidata contains about twice as many persons as DBpedia and YAGO. A value close to 0 for the overlap implies that DBpedia and YAGO contain almost no persons which are not contained in Wikidata. In conclusion, combining Wikidata with DBpedia or YAGO for a better coverage of the *Person* class would not be beneficial.

Notable exceptions are BabelNet and CaLiGraph, which often contain complementary instances. For example, DBpedia, BabelNet and CaLiGraph contain 1.2M, 2.4M, and 1.9M instances of the class *Person*, respectively, while DBpedia and BabelNet together are estimated to 2.9M, and all three together are estimated to contain even 3.9M instances of the class *Person*. The reasons for the high complementary of DBpedia/YAGO, BabelNet and CaLiGraph are their sources (only English Wikipedia vs. multiple lan-

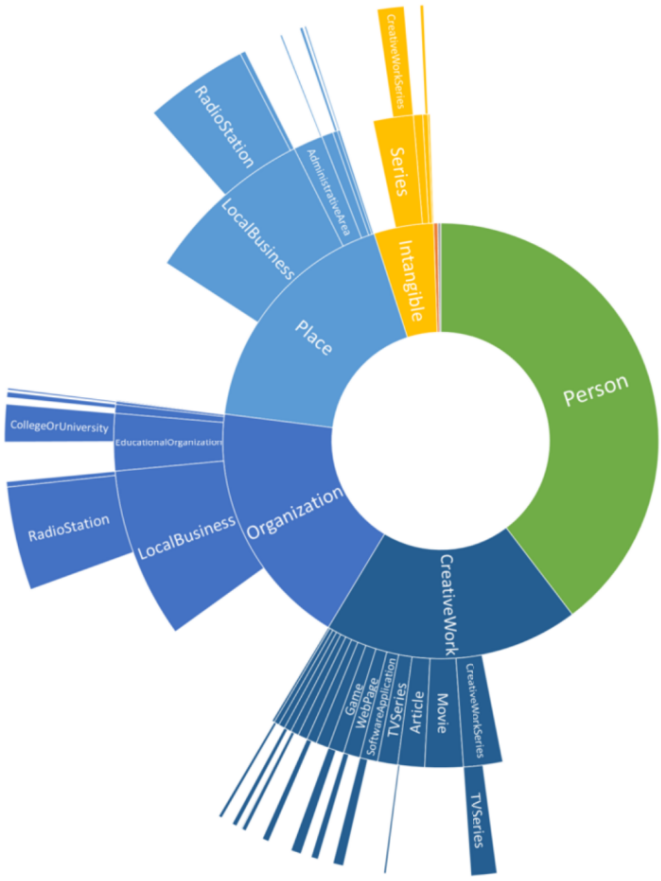

Figure 9. Instances in Voldemort

guage editions) and extraction mechanisms (especially the extraction from list pages in CaLiGraph, which leads to a larger number of instances overall).

Fig. 11 shows that the linkage between DBpedia, YAGO, BabelNet and CaLiGraph is mostly complete (i.e., most of the common instances are also explicitly linked). Since they are all generated from Wikipedia with different means, this is not much surprising. On the other hand, Nell, OpenCyc, and Voldemort have a much lower degree of linkage. This shows that links between KGs are only complete where they are trivial to create, and combining different knowledge graphs otherwise requires efforts in improving the interlinking as a preliminary step.

5. Conclusion and Outlook

In this chapter, we have given an overview of publicly available, cross-domain knowledge graphs on the Web. We have compared them according to different metrics which might be helpful to implement an eXplainable AI project in a given domain.

Besides the metrics used for this comparison, there are quite a few more which help in the selection and assessment of a given KG. For example, data quality in KGs has not

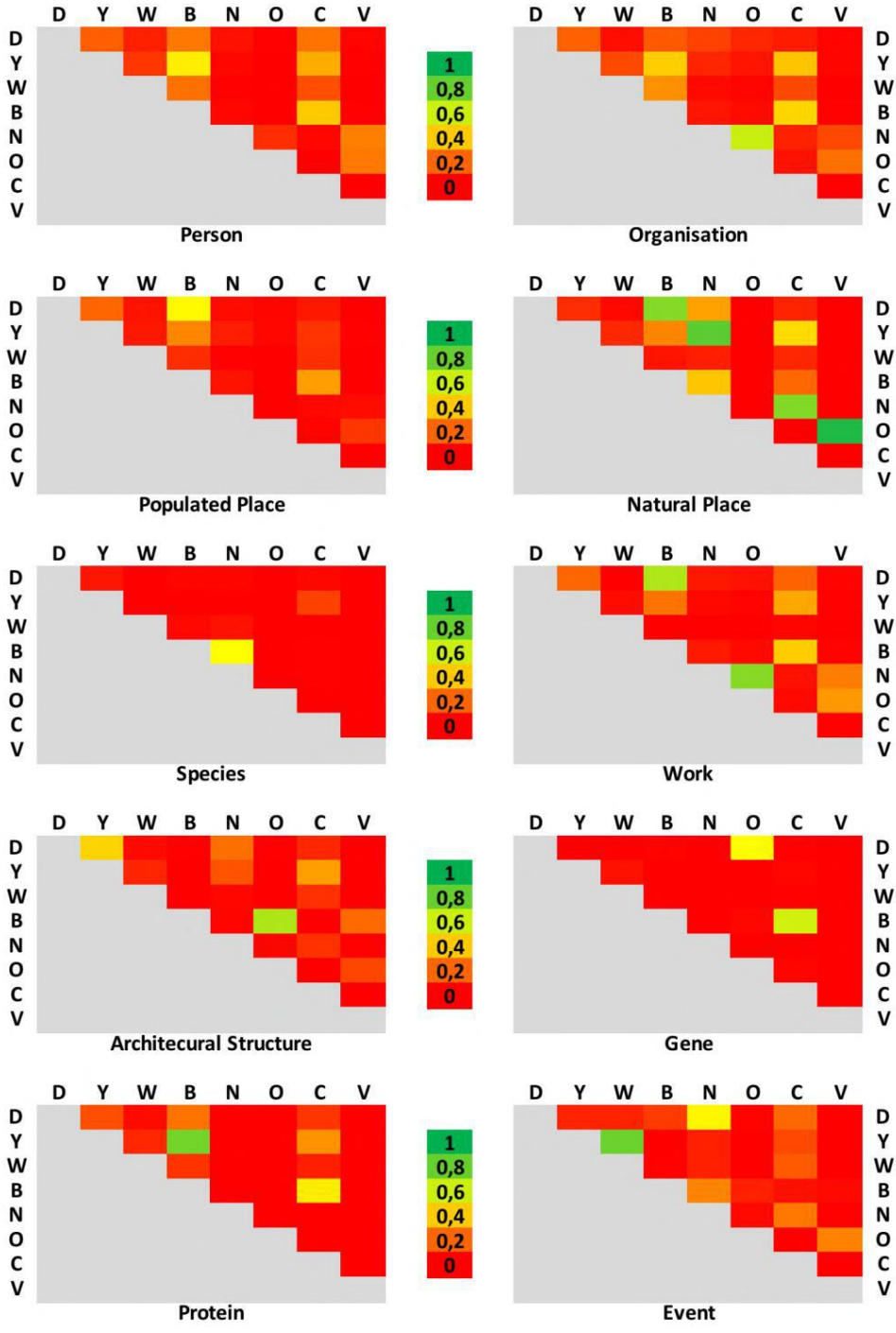

Figure 10. Fraction of entities in a pair of knowledge graphs which is *not* contained in the larger of the two graphs.

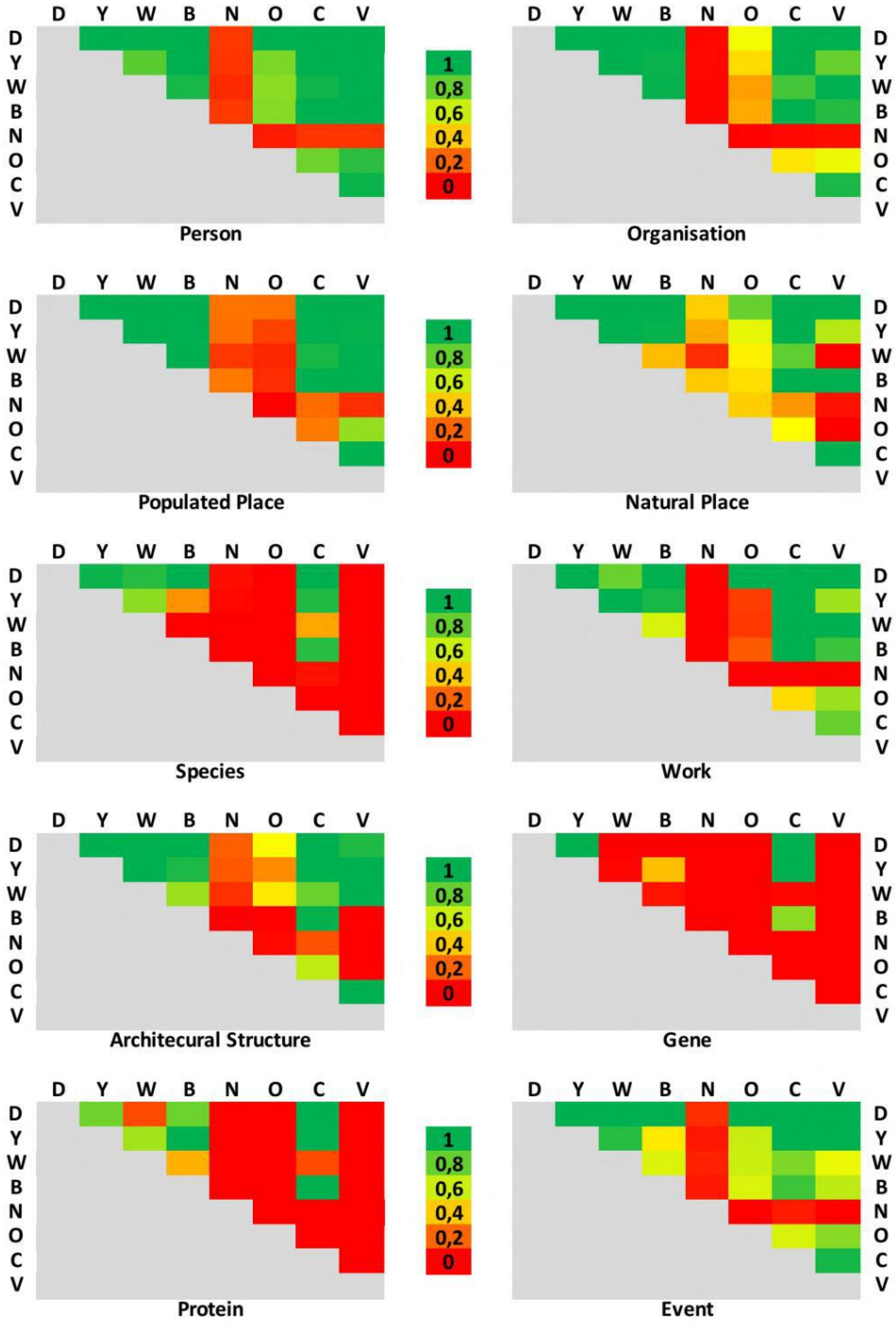

Figure 11. Existing entities in two KGs in relation to the number of links.

been considered in this chapter, since there are already quite elaborate surveys covering this aspect [6,44].

So far, we have measured the overlap of knowledge graphs only based on entities. Another helpful metric would be the overlap on the statement level. Even if two knowledge graphs cover the same entity, the information they contain about that entity might still be complementary. For example, for the entity *University of Mannheim*, DBpedia has the exact number of undergraduate students, PhD students, etc.[10], while Wikidata lists all faculties[11] and can provide a list of researchers employed at the university[12]. The density of information differs as well: while YAGO lists 3 alumni of the University of Mannheim[13], DBpedia lists 11 and Wikidata even 85 alumni[14]. Even contradicting information can be found [2]: for example, DBpedia and Wikidata provide a different number of students and Wikidata and YAGO provide different founding dates of the University of Mannheim.

Developing cross-domain knowledge graphs is an active field of research, and new developments emerge every once in a while. They differ in the data they use and/or the method of extraction:

- *DBkWik* [14,15,16] uses the extraction mechanism of DBpedia and applies it to a multitude of Wikis. The intermediate result is a collection of a few thousand isolated knowledge graphs, which have to be integrated into a coherent joint knowledge graph.
- *Chaudron* [39] uses Wikipedia as a source and focuses on quantifiable values (e.g., sizes, weights, etc.). Besides the mere extraction, Chaudron uses sophisticated methods for recognizing and converting units of measurement.
- The Linked Hypernym Dataset (*LHD*) [17], like the aforementioned WebIsALOD, focuses on the extraction of a hypernym graph. It uses a deep linguistic analysis of the first paragraph in Wikipedia.
- *ClaimsKG* [41] extracts claims from fact checking Web pages, such as politifacts, and interlinks them with other knowledge graphs such as DBpedia, which also allows for finding related claims.

The methods discussed in this chapter can be used to assess those emerging knowledge graphs and discuss their added value over existing ones. So, for example, for the above mentioned DBkWik, we have shown that it is highly complimentary to DBpedia: 95% of all entities in DBkWik are not contained in DBpedia and vice versa.

In summary, knowledge graphs are a useful ingredient to XAI systems, as they provide ready-to-use cross-domain knowledge. With this chapter, we have given an overview of existing knowledge graphs on the Web, and some guidelines on picking one or more such graphs to build an application for a task at hand.

[10] http://dbpedia.org/page/University_of_Mannheim
[11] https://www.wikidata.org/wiki/Q317070
[12] https://w.wiki/7UU
[13] https://bit.ly/2U4wL0A
[14] https://w.wiki/7UV

References

[1] Wouter Beek, Joe Raad, Jan Wielemaker, and Frank Van Harmelen. sameas. cc: The closure of 500m owl: sameas statements. In *European Semantic Web Conference*, pages 65–80. Springer, 2018.

[2] Volha Bryl, Christian Bizer, Robert Isele, Mateja Verlic, Soon Gill Hong, Sammy Jang, Mun Yong Yi, and Key-Sun Choi. Interlinking and knowledge fusion. In *Linked Open Data–Creating Knowledge Out of Interlinked Data*, pages 70–89. Springer, 2014.

[3] Bruce G Buchanan. A (very) brief history of artificial intelligence. *Ai Magazine*, 26(4):53–53, 2005.

[4] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110, 2010.

[5] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48, 2016.

[6] Michael Färber, Basil Ell, Carsten Menne, Achim Rettinger, and Frederic Bartscherer. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web (to appear)*, 2016.

[7] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 7(3):46–76, 2011.

[8] Mikhail Galkin, Sören Auer, and Simon Scerri. Enterprise knowledge graphs: A backbone of linked enterprise data. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 497–502. IEEE, 2016.

[9] Jose Manuel Gomez-Perez, Jeff Z Pan, Guido Vetere, and Honghan Wu. Enterprise knowledge graph: An introduction. In *Exploiting linked data and knowledge graphs in large organisations*, pages 1–14. Springer, 2017.

[10] Harry Halpin, Patrick J Hayes, James P McCusker, Deborah L McGuinness, and Henry S Thompson. When owl: sameas isn't the same: An analysis of identity in linked data. In *International semantic web conference*, pages 305–320. Springer, 2010.

[11] Nicolas Heist and Heiko Paulheim. Uncovering the semantics of wikipedia categories. In *International semantic web conference*, pages 219–236. Springer, 2019.

[12] Nicolas Heist and Heiko Paulheim. Entity extraction from wikipedia list pages. In *Extended Semantic Web Conference*, 2020.

[13] Sven Hertling and Heiko Paulheim. Webisalod: providing hypernymy relations extracted from the web as linked open data. In *International Semantic Web Conference*, pages 111–119. Springer, 2017.

[14] Sven Hertling and Heiko Paulheim. Dbkwik: A consolidated knowledge graph from thousands of wikis. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 17–24. IEEE, 2018.

[15] Sven Hertling and Heiko Paulheim. Dbkwik: extracting and integrating knowledge from thousands of wikis. *Knowledge and Information Systems*, pages 1–22, 2019.

[16] Alexandra Hofmann, Samresh Perchani, Jan Portisch, Sven Hertling, and Heiko Paulheim. Dbkwik: Towards knowledge graph creation from thousands of wikis. In *International Semantic Web Conference (Posters, Demos & Industry Tracks)*, 2017.

[17] Tomáš Kliegr and Ondřej Zamazal. Lhd 2.0: A text mining approach to typing entities in knowledge graphs. *Journal of Web Semantics*, 39:47–61, 2016.

[18] Freddy Lecue. On the role of knowledge graphs in explainable ai. *Semantic Web Journal (Forthcoming). http://www. semantic-web-journal. net/content/role-knowledge-graphs-explainable-ai*, 2019.

[19] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6(2), 2013.

[20] Douglas B Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[21] Robert Meusel, Christian Bizer, and Heiko Paulheim. A web-scale study of the adoption and evolution of the schema. org vocabulary over time. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 15. ACM, 2015.

[22] Robert Meusel and Heiko Paulheim. Heuristics for fixing common errors in deployed schema. org microdata. In *European Semantic Web Conference*, pages 152–168. Springer, 2015.

[23] Robert Meusel, Petar Petrovski, and Christian Bizer. The webdatacommons microdata, rdfa and microformat dataset series. In *International Semantic Web Conference*, pages 277–292. Springer, 2014.

[24] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[25] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[26] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

[27] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62(8):36–43, 2019.

[28] Heiko Paulheim. What the adoption of schema. org tells about linked open data. In *Joint Proceedings of USEWOD and PROFILES*, 2015.

[29] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[30] Heiko Paulheim. How much is a triple? estimating the cost of knowledge graph creation. In *ISWC 2018 Posters and Demonstrations, Industry and Blue Sky Ideas Tracks*, 2018.

[31] Heiko Paulheim and Simone Paolo Ponzetto. Extending dbpedia with wikipedia list pages. *NLP-DBPEDIA@ ISWC*, 13, 2013.

[32] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428, 2016.

[33] Daniel Ringler and Heiko 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)*, pages 366–372. Springer, 2017.

[34] Petar Ristoski and Heiko Paulheim. Semantic web in data mining and knowledge discovery: A comprehensive survey. *Web semantics: science, services and agents on the World Wide Web*, 36:1–22, 2016.

[35] Dominique Ritze and Heiko Paulheim. Towards an automatic parameterization of ontology matching tools based on example mappings. In *Proc. 6th ISWC ontology matching workshop (OM), Bonn (DE)*, pages 37–48, 2011.

[36] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer, 2014.

[37] Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. A large database of hypernymy relations extracted from the web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 360–367, 2016.

[38] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[39] Julien Subercaze. Chaudron: extending dbpedia with measurement. In *European Semantic Web Conference*, pages 434–448. Springer, 2017.

[40] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *16th international conference on World Wide Web*, pages 697–706, 2007.

[41] Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. Claimskg: A knowledge graph of fact-checked claims. In *International Semantic Web Conference*, pages 309–324. Springer, 2019.

[42] Alberto Tonon, Victor Felder, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Voldemortkg: Mapping schema. org and web entities to linked open data. In *International Semantic Web Conference*, pages 220–228. Springer, 2016.

[43] Denny Vrandečić and Markus Krötzsch. Wikidata: a Free Collaborative Knowledge Base. *Communications of the ACM*, 57(10):78–85, 2014.

[44] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges 23
I. Tiddi et al. (Eds.)
IOS Press, 2020

doi:10.3233/SSW200010

Foundations of Explainable Knowledge-Enabled Systems

Shruthi CHARI [a], Daniel M. GRUEN [b], Oshani SENEVIRATNE [a], and Deborah L. MCGUINNESS [a]

[a] *Rensselaer Polytechnic Institute, Troy, NY, USA*
[b] *IBM Research, Cambridge, MA, USA*

Abstract. Explainability has been an important goal since the early days of Artificial Intelligence. Several approaches for producing explanations have been developed. However, many of these approaches were tightly coupled with the capabilities of the artificial intelligence systems at the time. With the proliferation of AI-enabled systems in sometimes critical settings, there is a need for them to be explainable to end-users and decision-makers. We present a historical overview of explainable artificial intelligence systems, with a focus on knowledge-enabled systems, spanning the expert systems, cognitive assistants, semantic applications, and machine learning domains. Additionally, borrowing from the strengths of past approaches and identifying gaps needed to make explanations user- and context-focused, we propose new definitions for explanations and explainable knowledge-enabled systems.

Keywords. Knowledge Graphs, eXplainable AI, Explainable Knowledge-Enabled Systems, Historical Evolution

1. Introduction

The growing incorporation of Artificial Intelligence (AI) capabilities in systems across industries and consumer applications, including those that have significant, even life-or-death implications, has led to an increased demand for explainability. To accept and appropriately apply insights from AI systems, users often require an understanding of how the system arrived at its results.

Such an understanding can include having a model of how the underlying AI system operates, how it was constructed, and how the data used to develop and train it matches the situations in which it was used. It can include information about the specific features of the current situation that contributed to the system's determination. It can also include descriptions of the underlying rationales and reasoning paths the system used to arrive at a conclusion, which in turn can be based on observed statistical regularities, models of underlying mechanisms and causal relationships, and temporal patterns. We draw a distinction, between *transparency*,[1] by which we mean general information about a system's operation, capabilities, underlying training data, and fairness, and *explainability*,

[1] In this chapter, we use quotes for terms that we introduce or for direct quotations from publications, and we use italics to either emphasize terminology from papers or highlight important terms.

by which we mean the ability of a system to provide information describing and justifying how a specific result was determined along with the overall context. We build on this notion of explainability and present desired properties for explanations and redefine explanations supporting a user's perspective in Section 2.

By their very nature, explanations are user focused; explanations are needed because they provide information that would otherwise be absent that helps a user trust, apply, and maximally benefit from the AI system's operation. Thus, the need for explanation, and the types of explanations required, are contextual, depending on users, their roles, their prior knowledge, and the situation. For example, a physician recommending a non-standard treatment regimen might want to understand what aspect of the current patient's condition led to an unexpected result, and how the reasoning behind it aligns with scientific knowledge about biological and pharmacological mechanisms. A patient-facing explanation for the same result may need to include more basic information on the condition and what is unique about the patient's situation. An explanation aimed at a hospital administrator or insurance coordinator may need to include information about potential biases that could lead to a lack of fairness in the recommendation.

Explanations can have deeper value beyond the "gating" role they play in helping users determine which results should be trusted and applied. Explanations provided in the above example could contribute to the mental model the physician is constructing of the patient, and of diseases and biological mechanisms in general, that could be valuable in future treatment decisions they make for that patient and others. Explanations also contribute to the model users are creating of the system itself, by exposing the kinds of information and processing mechanisms the system utilizes. Norman famously described how users construct mental models of systems with which they interact across gulfs of execution and interpretation [58]. With AI systems, explanations can help users simultaneously construct models of the system with which they are interacting, and of the underlying domain and situation in which the system is being used.

The importance of explainability is particularly salient with collaborative AI systems meant to work in tandem with human users to augment rather than supplant their skills and capabilities. A "Distributed Cognition" approach [35] is informative here, in which cognition is seen to take place not within the head of any one individual, but rather through the exchange and transformation of representations across multiple actors and artifacts [38]. The ability for a system to provide explanations, and respond to queries that reference other information relevant to the situation, expands the range of ways in which the system and human actors can interact.

1.1. Historical Evolution

Explainability has been a major goal since the early days of AI. In this chapter, we focus on the broad class of knowledge-enabled systems, instead of simply knowledge-based systems. We include rule-based systems as well as hybrid AI systems that may include a wide range of reasoning components including potentially inductive or abductive reasoning as well as the more traditional deductive reasoning. As such, we include historical explanation work (e.g., [13,68,70]) and also explanation work aimed more at evolving hybrid AI systems (e.g., [27,54,66]). The survey includes the domains of expert systems, cognitive assistants, Semantic Web [32], and, more recently, explanations that work with black-box models, i.e., deep learning models [44]. With this background, we will now present a historical perspective on the evolution of explainable AI.

Many early AI systems took a rule-based, expert system approach. Expert systems (e.g., [68,70]) were inherently explainable in that they used a set of rules to come to conclusions, so explanations could be generated that provided a detailed or abstracted collection of the rule executions as an explanation of a conclusion. During the expert system era, much work focused on explaining these systems and their decisions to the end-user. Explanations were broadly intended to address the *Why*, *What*, and *How* aspects of an AI system that produces a result. Dhaliwal et al. [16] provide an overview of these explanation types and state that the *Why* explanations were populated with the justification for a conclusion, the *How* explanations contained a trace of the mechanistic functioning of the system, and, the *What* explanations exposed the system's decision variables involved in the conclusion. Explanations produced by these systems were mainly focused on introducing the rationale behind a system's decision and the way the system works. Additionally, while trace-based explanations produced by expert systems captured the *why* and *how* aspects, they typically did not account for the context of a user when they generated explanations. There were a wide range of expert systems early on. For example, MYCIN [67] was an early expert system that supported medical diagnosis using a rule-based inference engine and included a trace-based explanation component.

Today, with the availability of vast volumes of data, deep learning algorithms are being widely used. However, these models are largely uninterpretable, and a significant focus of explainable AI research (e.g., DARPA XAI Report 2017 [27]) is focused on explaining the underlying mechanisms of these black-box models. In our opinion, gaining transparency into the black boxes can be useful, and it may decrease the "unintelligibility aspect" [47]. However, it is not enough to provide *personalized, tailored, and trustworthy* [36] explanations to consumers of AI models. Additionally, machine learning (ML) models often output a score or probability as predictions. While the number may be useful to understand some level of confidence, a single number lacks context, and thus is often inadequate without additional information. Semantic Web representation and reasoning work is well suited to help here. Standards for representing terminology, e.g., RDF and OWL [55], as well as representing provenance (e.g., PROV-O [42] and nanopublications [26]), have emerged and can be used to encode information along with its provenance and a system's reasoning provenance, and this may be used to augment explanations. The inclusion of provenance into the underlying representation and thus potentially into the explanations partially addresses the *What* and *Why* aspects of the reasoning behind presenting explanations to consumers.

Recently, researchers have acknowledged an increasing need to include explainability modules into AI systems. As a consequence, several survey papers [10,36,57] have highlighted past noteworthy efforts in explainable AI.[2] These survey papers emphasize the fact that different situations, users, and contexts demand different kinds of explanation [3]. Different AI systems are geared toward addressing an explanation type (or rarely, a combination), e.g., expert systems typically provide trace-based explanations, deep learning models can be leveraged to offer contrasting explanations, etc. We believe that the next-generation AI systems need to go beyond the *Why, What, How* aspects and produce explanations that additionally prioritize issues related to the setting, users' understanding, and contexts. At a minimum, these AI systems need to include a provenance

[2]We choose to use the explainable AI phrasing for explainability efforts in AI as XAI has come to be associated with the DARPA eXplainable AI (XAI) program. Our focus is broader than the program's focus on explaining and interpreting black box ML methods from a cognitive perspective.

component to support trust and provide users with tools that can access a reasoning trace to further explore or serve as a means of understanding. In Section 3, we will review a few of these past approaches mentioned in this section.

1.2. Shift and Current Focus of Explainable AI

The evolution of AI systems has been heavily influenced by the availability of resources, computing power and data. As previously stated, AI has moved from primarily using rule-based expert systems to using ML methods, and, sometimes, hybrid methods. With these changes, there has been a shift in the focus of explainability, due to the new challenges of the interpretability of complex ML models. The initial explanation focus on working from system traces to provide a notion of what was done has expanded to including a focus on including a notion of interpretability of an underlying ML model. This *interpretable* ML work may be a first generation of explanation of ML, but as we will expand on below, more is needed. Additionally, in the Semantic Web [32], and more generally, in knowledge representation-based applications, the focus has expanded from traditional *What* explanations to include explanations addressing information attribution and provenance aspects. The motivations for those expansions include improving the trustworthiness of information being represented in knowledge graphs (KGs), and further, to provide more context for users as they are deciding how to use the information in analysis applications. Further, AI models are now being employed in user-facing settings where there is a need for personalized conclusions. Hence, there is a need to rethink explanations produced by AI systems from a user perspective and include components to educate users, align with their cognitive model, help them trust the system, provide relevant information, and tailor suggestions to a user's contexts [10,57]. Borrowing strengths from explanations provided by past approaches, we will attempt to present, synthesize, and refine a definition of explanation and explainable knowledge-enabled systems with an acknowledgment of desired explanation properties that fit today's settings.

2. Terminology

Several researchers have proposed comprehensive definitions of explanations [18,23,54, 71] and have presented explanation components that they deem necessary to satisfy either their work or the domains where they hope the explanations will be useful. However, with a shift of focus in AI we feel the need to revisit the work on defining explanation as we consider what is desirable in next-generation "explainable knowledge-enabled systems." In this section, we list desirable properties (Section 2.1) for both explanations and explainable knowledge-enabled systems that generate these explanations, and use these properties as a basis to provide definitions.

2.1. Desirable Properties

2.1.1. Explanations

As a part of our list of desirable properties for explanations, we present properties, such as, *improving user appeal, and achieving user understandability*, that have been explored as explanation components in the past, and that will be useful in designing explanations

suitable for the end-user. In addition, we propose including a higher priority on the inclusion of features, such as, *provenance* and *adapting to user's context* that will have a renewed focus in making explanations user-centric and in mitigating the unintelligibility aspect of current ML methods. We are aligning with others who have called for a greater user focus in explanations [45,56,64].

- **Be understandable:** Borrowing from desired properties of explanations stated by Swartout and Moore [71], we highlight that for explanations to be *understandable* by the user, the explanations should use terminology familiar to the user. If terminology is potentially unfamiliar, then we also suggest that capabilities be included for obtaining definitions of terms, thereby *educating* users. Understandability has the potential to be significantly increased if the AI system incorporates *user feedback* and a model of *user context*.
- **Include provenance:** *Provenance* is a property of explanations that has either been absent in some past descriptions of explanations [71], or has not had the emphasis that it deserves now. As systems expand to include more diverse content the need for capturing provenance increases. Explanations need to include *provenance* that includes information about the domain knowledge utilized by the system, along with the methods used to obtain that knowledge. We borrow from the "counterfactual faithfulness" idea proposed by [18], and argue that, as part of the *provenance* components, explanations need to carry *causal information* about the conclusion, if present in domain literature or supported by expert knowledge.
- **Appeal to user:** Paraphrasing Swartout and Moore [71], we note that explanations need to be *rich, coherent, and appeal to the user*.We propose that explanations need to expose facts that the user finds *resourceful and sufficient for further exploration*. A *resourceful* explanation contains enough granular content and evidence to appeal to the user's mental cognition and current needs. A *sufficient* explanation contains content that the user requires to carry out their tasks. A subtlety in generating explanations that appeal to the user would be to tailor the explanation length to the user's needs and preferences, i.e., to avoid lengthy explanations with content that might not be useful to the user or that they already understand. Further, we acknowledge that the resourcefulness and sufficiency aspects of explanations might be hard to measure in real-time. However, we suggest that explainable knowledge-enabled systems should be designed after an analysis of user requirements and utilize techniques to employ dynamic and static evaluation strategies to help realize these goals. More specifically, dynamic strategies could involve interactive mechanisms, such as the delivery of persuasive messages used by Maimone et al. [48], and static evaluation strategies could include user surveys conducted to evaluate the effectiveness of the systems, such as the one by Glass et al. [25].
- **Adapt to users' context:** Besides being user-centric, explanations need to be tailored to the user's *current scenario and context*. Explainable AI systems not only need to leverage information about the user (as may have been captured in a user profile [65,69]), but they also need to identify the user's intent and adapt the *explanation form* to connect to the user's mental model and align with the user's intent. For example, an explanation may include a contrastive hypothesis that relates to the user's intent or statistical evidence to provide more support to enhance a user's belief. In a later chapter, "Directions for Explainable Knowledge-enabled

Systems," in this book, we present different explanation types and their various focii that would allow AI systems to generate diverse explanations.

Overall, explanations should serve beyond their original aim to teach [70], and provide *trustworthy, transparent, unambiguous accounts* of automated tasks to end-users.

2.1.2. Explainable Knowledge-Enabled Systems

While many have attempted to define explanations (e.g., [17,71]), additional efforts have attempted to improve the generation of explanations (e.g., [25,39,54]) and tackle various aspects of explainability (e.g., [61,57]). To begin to address the need of building explainable, knowledge-enabled AI systems, we present a list of desirable properties from the synthesis of our literature review of past explanation work. Our review primarily spans knowledge representation in expert systems [71], provenance and reasoning efforts in the Semantic Web [26], user task-processing workflows in cognitive assistants [53,54], and efforts to reduce unintelligibility in the ML domain [3,23,27]. Additionally, we analyzed explanation requirements from current literature, answering an increased need for *user-comprehensibility* [43], *accountability* [17] and *user-focus* [57]. In our literature review in Section 3 we will highlight approaches that exhibit these properties.

- **Modularity**: A modular design, such as, the one proposed by Swartout and Moore [70], is desirable, as it would allow systems to adapt models and functioning to users' requirements and scenarios. This property would also allow for the AI system to include explanation facilities that tap into various modules to expose information requested by and conducive to the user's needs.
- **Interpretability**: Borrowing from Mittlestadt et al. [57] and Hasan and Gandon [31], we believe that the interpretability of explainable knowledge-enabled systems enables them to be transparent, lending to the ability to provide trace-based accounts of their working. Additionally, we utilize Gilpin et al's definition of interpretability as a "science of comprehending what a model did." However, if the models used in the system are not interpretable, we propose that they should consider including proxy methods to be interpretable, for example, utilizing linear proxy models proposed by Gilpin et al. [23] that serve as a simplified proxy of the full model.
- **Support provenance:** Paraphrasing from the explanation requirements suggested by Hasan and Gandon [31], we agree that explainable knowledge-enabled systems should store the provenance of the information that their models rely on beyond just metadata. We believe that the inclusion of provenance aids AI systems in generating *resourceful and sufficient* explanations for users, providing them with resources for further exploration.
- **Adapt to user's needs:** We propose that AI systems need to be *adaptive* and *interactive*, adapting their functioning and explanation generation capabilities to suit the user's requests and contexts. To this end, and to provide tailored explanations, Ribera and Lapedriza [64] have identified user categories (domain experts, AI researchers, and lay users) and presented their contrasting demands from an AI system. Further, the ability to be adaptive would be enhanced by a modular design, as suggested earlier, and would aid the system in generating explanations in various forms to suit the user's understanding and their needs.

- **Include explanation facilities:** Inspired by McGuinness et al.'s cognitive assistants explanation frameworks [53,54], we propose that the design of the explanation facilities should be addressed early and in detail in the design phase, to ensure that the AI system is capable of supporting the requirements of the explanation facilities within its design. Explanation facilities could constitute a wide-range of user-facing interfaces, such as, dialogue systems, visualizations, and feedback systems that the user interacts with and provides feedback to the AI system about the explanations generated or a need for further clarifications. Hence, since explanation facilities would require additional information, such as, provenance, and would need the system to incorporate feedback and adapt to context, we recommend that their design be coupled with the AI system design.
- **Include/Access a knowledge store**: We recommend that explainable knowledge-enabled systems store the *domain knowledge* they rely on, the *user's mental model* they appeal to, and the *explanation components* they are generating. Additionally, we relax the inclusion of knowledge in that an AI system might provide access to a knowledge store - as the system may host it, or it may use some other system's hosting and contribute to and access that store. By knowledge store, we refer to data storage mechanisms (KGs or semantic representations are preferred) that can store knowledge of various forms spanning categories such as background knowledge, domain knowledge, etc.
- **Support compliance and obligation checks:** In addition to hosting/accessing knowledge stores, we recommend that explainable knowledge-enabled systems store an encoding set of expert knowledge in their field of application. These encodings should be sufficient to determine if the system complies with the standards and practices in that field. Additionally, we also recommend that explainable knowledge-enabled systems attempt to adhere to standards for the proposed explainable AI models, such as [3,23]. Furthermore, we suggest that compliance and obligation checks be evaluated on the system post-construction.

2.2. *Definitions*

Having identified desirable properties for explanations and explainable knowledge-enabled systems, will now provide a set of definitions leveraging our review of the explanation literature and our analysis of the current AI landscape. Our goal is to reflect the needs of explainable AI in current times and provide a summary of the desirable properties to achieve better explainability.

2.2.1. *Explanation*

We define an explanation in the computational world as, "an account of the system, its workings, the *implicit and explicit* knowledge used in its reasoning processes and the specific decision, that is *sensitive* to the end-user's *understanding, context, and current needs*."

2.2.2. *Explainable Knowledge-Enabled Systems*

We define "explainable knowledge-enabled systems" to be, "*A*I systems that include a representation of the domain knowledge in the field of application, have mechanisms

to incorporate the *users' context*, are *interpretable*, and host *explanation facilities* that generate *user-comprehensible, context-aware, and provenance-enabled* explanations of the mechanistic functioning of the AI system and the knowledge used."

3. Approaches

We present past approaches that have addressed various aspects of explainability related to trust, transparency, provenance and interpretability. To the extent possible, we group publications by technical domain: knowledge-based systems, Semantic Web applications, cognitive assistants, and ML systems, in an attempt to show the progression of methods within those domains. In Section 3.1, we consider work from the 1970s-1990s that sought to utilize the trace explainability strengths of rule-based systems to explain the process used to arrive at decisions. In Section 3.2, we review provenance and explanation modeling efforts and posit them as contributors to the development of trustworthy and explainable semantic applications. In Section 3.3, we focus on efforts to explain task-based workflows in personal assistants and intelligent tutoring settings. We end with a review of papers that improve the interpretability and trust aspects of ML methods in Section 3.4. While each of these vast domains has large volumes of published literature, we restrict ourselves to seminal work on explainability in the domain or publications that have introduced novel techniques to tackle different aspects of explainability. As a conclusion of each domain subsection, we provide a brief summary of the methods utilized to address explainability and describe any lessons applicable for the development of future explainable AI methods.

Table 1 contains an evaluation of the foundational AI systems, reviewed against the criteria we defined for explainable knowledge-enabled systems (Section 2.1). The chronological order allows us to view trends in explainability over the years and also helps expose shifts in the areas of focus and strengths of the class of approaches. We observe that explanations were well-explored as a topic of interest in the AI community from the early 1990s - mid-2000s. We note that, even within the expert systems era, the AI architecture evolved from simply generating trace-based accounts of decisions to including modular explanation facilities ([11,13,70]) that sometimes could produce provenance-enabled ([13]), adaptive and user-customizable ([60]) explanations. Additionally, observe that, among other classes of approaches in our review, explanations have been best established in cognitive assistants, which also have the most direct impact on human decision-making capabilities. However, we notice that, with more recent systems in the Semantic Web and ML domains, there has been a shift in explainability from building explanation facilities to minimally ensuring that AI models are interpretable [63] and support provenance [50] for further tracing. Further, most AI systems in our review ([11,13,50,53,54,68,70,72]) satisfy the 'Compliance Checks' criteria by leveraging logical rule-based or other deductive reasoners to check or enforce compliance. Also, systems such as the Disciple-LTA and Common Ground Learning and Explanation (COGLE) deployed in critical settings of military and aviation, respectively, have features to allow both expert and lay users to provide feedback about the system's explanations and outputs. Hence, indicating that system supported features are partially driven by the domain of application. Finally, while our evaluation was conducted on a carefully selected set of approaches, our findings on explainability trends are in-line with a larger,

Table 1. Foundational explanation approaches and desired features of explainable knowledge-enabled systems

Year	System	Application	Knowledge Store	Interpretable	Explanation Facilities	Modular	Adaptive	Support Provenance	Compliance Checks
1977	MYCIN [68]	Healthcare	✓	✓					✓
1982	NEOMYCIN [13]	Healthcare	✓	✓	✓	✓		✓	✓
1989	CLASSIC [11]	General Purpose	✓	✓	✓	✓	✓		✓
1991	Explainable Expert System (EES) [70]	Program Advisor	✓	✓	✓	✓			✓
2004	Inference Web Framework [53]	General Purpose	✓	✓	✓	✓	✓	✓	✓
2005	Disciple-LTA [72]	Military	✓	✓	✓	✓	✓	✓	✓
2007	Integrated Cognitive Explanation Environment (ICEE) [54]	Office Assistant	✓	✓	✓	✓	✓	✓	✓
2009	Automated Policy Reasoning [39]	Judicial	✓	✓	✓		✓	✓	✓
2016	Local Interpretable Model-agnostic Explanations (LIME) [63]	General Purpose		✓	✓	✓			
2017	ReDrugs [49]	Medicine	✓	✓				✓	✓
2017 (ongoing)	Common Ground Learning and Explanation (COGLE) [28]	Unmanned Aircrafts	✓		✓	✓	✓		✓

systematic review conducted by Nunes and Jannach [59], who noted that explainability was best explored in the expert systems and cognitive assistants domains.

3.1. Knowledge-based systems

The 80s decade saw the rise of knowledge-based and expert systems, that were designed to assist humans where human resources were limited [16]. Expert systems and knowledge-based systems both contained an encoding of knowledge. More specifically, in the case of expert systems, the knowledge encoded was that of expert's knowledge, typically in the form of rules. In our review, we will not make distinctions between these two classes of systems and will focus on identifying the explainability components of these systems. From an implementation perspective, both of these systems required the engineering and encoding of multiple rules to support inference. This reliance on rules made these systems inherently explainable, as one could trace back the rules to identify the factors that lead to a conclusion. Subsequently, researchers have introduced different types of explanations [13,68], and approaches to improve explanation generation [11], and to introduce more granular content into explanations generated by these systems [70].

3.1.1. Early Expert Systems: MYCIN and NEOMYCIN

The MYCIN [67,68] paper was one of the first to introduce computer-based explanations, and, is regarded as a foundational and seminal work. The goal of the MYCIN system

was to identify highly probable carriers of infectious diseases, and suggest treatments for the diseases. The system provided explanations by exposing the inference trace that lead to a decision. The system was able to trace back and expose the reasoning, that served as justifications of decisions. In particular, MYCIN provided *Why* and *How* explanations [16]. The *Why* explanations included facts and task-based information to address a user's queries. The *How* explanations explained the manner and trace in which the system generated the conclusion.

To enhance the *Why* and *How* explanations, a descendant of MYCIN - NEOMYCIN [13] produced strategic explanations comprised of meta-knowledge and the problem-solving strategies to adapt the MYCIN system to a teaching setting. NEOMYCIN built on MYCIN's inability to explain beyond the expert knowledge known to the system and added a component that leveraged explicit encodings of problem-solving strategies used to generate the medical knowledge for use in its explanations. To this end, the NEOMYCIN system used a meta-strategy to decide what portion of the rules to invoke from data sources, including an etiological taxonomy, disease knowledge, and causal associations. The metastrategy contained rules that a human would use to undertake tasks such as building hypothesis, pursuing them, identifying problems, etc. In essence, the NEOMYCIN system attempted to mimic human decision-solving, where one would eliminate a hypothesis based on the search space, and not by merely navigating the knowledge ("bridge concepts" [13]) that the system already holds. Further, NEOMYCIN introduced the idea of separating knowledge to make the system more accessible, which was further adopted by Moore and Swartout in their Explainable Expert System [70] effort, discussed later in this section. While the strategic explanations generated by NEOMYCIN are desirable, they might be onerous for user consumption due to a surplus of details.

3.1.2. Explainable Description Logics: CLASSIC

McGuinness and Borgida took an approach to explanation where each of the inferences that the underlying logical reasoning system could execute had a declarative explanation description and those individual explanation components could be used to build simple, complex, abstracted, or otherwise customized explanations [52]. Additionally, every expert rule that a knowledge-based system builder encoded in the system included a structured component that could be used to explain when that rule was used. These explanation "breadcrumbs" could then be used to assemble explanations when a user's actions triggered the execution of a rule.

The authors implemented their approach in the CLASSIC knowledge representation system, a description-logic-based language that provided a framework "to define structured concepts and make assertions about individuals in a knowledge base" [60]. The complete set of foundational inference rules that could be explained for the underlying description logic reasoner was also available for reuse in other systems [51]. This style of encoding axioms for every inference that a system could execute was also leveraged in the axiomatic semantics for other predecessors to today's description logic-based recommended language for encoding ontologies on the web: OWL [55]. The axioms for RDF, RDFS, and DAML+OIL were described in W3C Note[3] and then were used in a number of different reasoners to provide trace-based explanation capabilities.

[3]DAML+OIL axioms note link: https://www.w3.org/TR/daml+oil-axioms

3.1.3. Explainable Expert System

Moore and Swartout coined the term 'Explainable Expert Systems' (EES) in their widely-cited work [70]. The EES framework that aimed to provide explanations and was tested in a Program Enhancement Advisor setting. The explanations generated by the EES system borrowed from and had components of various knowledge sources including domain, problem-solving and system terminology. Further, the design of their EES system supported the generation of the various components of the explanations and were made of knowledge bases, a program writer, an explanation generator, an interpreter, and an execution trace. The EES system used a planning algorithm, wherein goals are reformulated if no viable match is found in the domain knowledge. The reformulation of goals was achieved by the representation of the domain knowledge into a concept hierarchy, via a language, such as, KL-ONE [12]. The EES framework was interactive in nature, and goals were reformulated based on user dialogue with the system. Additionally, users' queries were used as a cue to interleave domain and problem-solving knowledge traces into their explanations.

3.1.4. Summary

The explainable knowledge-based systems that we discussed introduced several types of explanations, including *Why, How, and Strategic* explanations (described earlier in Section 1.1). However, their reliance on encoding a large rule base makes them difficult to scale and extremely human-intensive to maintain. Today, we see the semi-automatic generation of rules and knowledge-base population via natural language processing and ontology-enabled extraction techniques. Many learnings from knowledge-based systems have been reused and expanded in the Semantic Web, as will be illustrated in Section 3.2.

3.2. Semantic Web

The creation of the World Wide Web (WWW) [8] made it possible to create content online and make existing content available online in digital formats. In their seminal paper, Berners-Lee, Hendler, and Lassila [9] state that the Semantic Web was intended to unify content being published online through tagging content with unique identifiers, or Uniform Resource Identifiers (URIs), representing the content utilizing well formed definitions from taxonomies and ontologies, and borrowing from the knowledge representation world to utilize structuring mechanisms for data. While these properties are desirable and necessary to enable data sharing and achieve a semantic understanding of digital content, they are not sufficient to make the content explainable to a broad range of users. However, the Semantic Web community has tackled the provenance aspect and trace-based aspects of explainability and developed several provisions to both include provenance in the semantic representation [26,42] and to supporting reasoning mechanisms, [34] to generate traces. As a direct consequence of the Semantic Web, the textual content is more accessible in knowledge graphs (KGs) via semantic representations [20]. Additionally, KG provisions have made it possible to provide justifications and provenance to suggestions. In this section, we will review some provenance encoding efforts and explainable semantic applications.

3.2.1. Provenance modeling efforts

There have been two somewhat recent foundational provenance efforts that paved the way for provenance-aware applications, namely the World Wide Web Consortium's work on a recommended standard for provenance on the web (PROV) with its associated encoding as an ontology PROV-O [42] and nanopublications [26]. Nanopublications provide a structure to associate triple statements with their provenance. In general, provenance is essential as it encodes information that can be used to explore where information came from and this information can be used to build trust in applications when they use this information to expose the evidence behind their recommendations.

3.2.2. Nanopublications

Nanopublications were conceived to help disambiguate and represent the context for scientific statements that were extracted from textual corpora and made available as triples. The authors identified that contextual information present in a document was imperative to understand a statement in relation to the full document. Hence, they designed nanopublications that provided a mechanism to associate metadata or annotations with statements. The schema of nanopublications has evolved over the years. In its current state,[4] nanopublications are composed of three named graph components, Assertions, Publication Information, and Provenance. The Publication Information graph stores metadata information about the creation of the content, or how it came to be, such as, the date of creation, author, etc. The Provenance graph contains metadata, such as, citation information. The assertion graph contains one or more subject-predicate-object statements with domain content.

Kuhn et al. [41] have proposed an Atomic, Independent, Declarative, and Absolute (AIDA) framework to encode atomic and indisputable assertion statements. They describe a metananopublication world in which nanopublications can be created from other nanopublications via different channels, for example, from authors creating content from scientific results, and from data mining algorithms generating nanopublications from existing unstructured data sources. Essentially through the metananopublications concept, the authors highlight that provenance can be interleaved and chained, to reflect the real world where multiple entities depend on each other at various levels of granularities.

3.2.3. The Provenance Ontology (PROV-O)

The PROV-O ontology [42] provides a formal mechanism to support comprehensive modeling of the provenance of digital objects. In their ontology, they support three primary forms of provenance contributors, agent-centered, object-centered, and process-centered forms. In PROV-O [5], provenance is modeled via three simple class types, i.e., 'entities'[6] which are generated by activities, and 'entities' and 'activities' that are 'associated with' and 'attributed to' agents, respectively. In the W3C note, the editors showcase the adequacy of the PROV-O ontology in modeling a use case where a blogger is exploring the provenance chain of a newspaper article while finding out who compiled the chart included in the article. The use case also illustrates that provenance needs to

[4] Nanopublication Guidelines: http://nanopub.org/guidelines/working_draft/
[5] PROV-O ontology W3C note: https://www.w3.org/TR/prov-o/
[6] Classes and properties are referred to by their label, and are enclosed within single quotes

be modeled comprehensively to ensure that users have a complete understanding of the information they are viewing.

There have been ontology alignment efforts on the PROV-O ontology to enhance usability and increase interoperability. These efforts include alignment of PROV-O with standard ontologies, such as, the TIME ontology, Semantic Sensor Network Ontology (SSN), and the Basic Formal Ontology (BFO). The PROV-O ontology has also served as a foundational ontology for several other provenance ontologies (e.g., Provenance for Clinical and Healthcare Research (ProvCare [73]) and Guideline Provenance Ontology (G-Prov [1])) that support provenance modeling in specific use cases with different levels of granularity.

3.2.4. *Provenance and Related Semantic Knowledge Graphs*

The Semantic Web community also allows for different alternatives of representing that information based on granularity and content needs such as named graphs, [7] reification, [8] etc., and there exist cross-domain open source KGs that host somewhat comparably rich provenance (e.g., Wikidata, [9] WebIsALOD [33]). Additionally, while we believe that provenance modeling is crucial to provide high-quality, trustworthy information to consumers, we acknowledge that it is not sufficient to capture user context or to personalize results. Recently, there has been an emergence of KGs that encode contextual and personal information [29,48], lending to the personalizing of semantic applications that are enabled by these KGs. Gyrard et al. [29] described the components of a personalized healthcare knowledge graph (PHKG) that are needed to monitor user health to help users combat chronic diseases, such as, asthma and obesity. In a similar effort, Maimone et al. developed Perkapp [48], a persuasive system that monitors people's lifestyles and persuades them to make healthier choices and stay on track. Their persuasive, knowledge-based system architecture contains a set of expert-generated rules and outputs persuasive context-aware messages to users based on their adherence to the rules.

3.2.5. *Reasoning Efforts*

We now present a selective overview of the reasoning efforts. We briefly introduced RDFS reasoning efforts in Section 3.1. RDFS reasoning results in justifications or trace-based accounts of *Why* a conclusion was made by the system, based on which rule fired. However, these justifications can be overwhelming for human consumption. To address this, Horridge et al. [37] proposed laconic and precise justifications that do not 1. conceal detail, 2. expose axioms that are relevant to the justification, and 3. are atomic, in that multiple fine-grained cores can be highlighted. Besides the laconic justification effort, there have been other efforts to improve explainability of justifications and we discuss one such effort, the AIR (Accountability in RDF or AMORD[10] [40] in RDF) language.

3.2.6. *Explanations for Automated Policy Reasoning*

AIR language that had a broader focus on modeling explanations serving to explain inference traces from policy reasoning. AIR is a Semantic Web-based rule language fo-

[7] Named Graphs: https://www.w3.org/2009/07/NamedGraph.html
[8] Reification: https://www.w3.org/TR/rdf-primer/#reification
[9] Wikidata: https://www.wikidata.org/wiki/Wikidata:Main_Page
[10] AMORD (A Miracle Of Rare Device) is an explanation system developed for MIT scheme in the 1970's

cused on generating and tracking explanation for inferences and actions [39]. The Massachusetts Institute of Technology (MIT) Decentralized Information Group developed the AIR language, as an extension to N3Logic [7] to support accountable privacy protection in Web-based information systems conforming to Linked Data principles. Accountability and privacy protection are enabled through auditable trace-based explanations. AIR supports *Linked Rules*, which can be combined and reused like Linked Data. Additionally, AIR explanations can be used for further reasoning.

AIR provides two independent ontologies. One ontology allows the specification of AIR rules,[11] and the other one allows describing justifications.[12] The reasoning steps of the AIR reasoner are considered as events and modeled as subclasses of `air:Event`. `air:Rule` represents rules, and it is defined as a subclass of `air:Operation`. The ontology also provides properties to enable representing variable mappings in the performed operations. AIR provides a means to write explicit explanations using the assertion property associated with rules. This property is composed of two components, `air:Statement`, which is the set of triples being asserted, and `air:Justification`, which is the explicit justification that needs to be associated with the statement.

Example policy reasoning with explanations using AIR:

Parts of the *Massachusetts Disability Discrimination Law* were translated into a computer interpretable policy using AIR. A user's phone records requesting some service and subsequently getting denied based on his disability recorded in the phone logs were captured in RDF. Once the AIR reasoner is invoked with the policy file, and the phone log in RDF, a user can visualize the annotated transaction log that contains the reasoning output. Figure 1 contains a partial proof tree with natural language assertions.

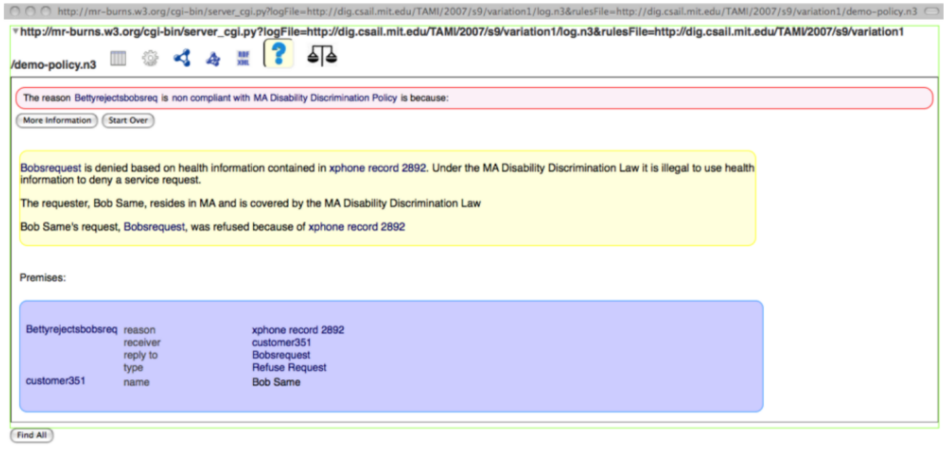

Figure 1. AIR Justification or Explanations View: Once a user clicks the "Why?" button, they will see a description appear in the "Because" box, and the premises that support the justification appear in the "Premises" box. When the user clicks the "More Information" button, the descriptions corresponding to outer rules in the proof tree will be appended to the "Because" box, and the "Premises" box is overwritten with the corresponding set of premises in the proof tree. When all the descriptions in the proof tree have been traversed, the message "No more information is available from the reasoner" will be displayed in the "Premises" box. At any given time, this proof exploration can be restarted by clicking the "Start Over" button. [Image taken from website [13]]

[11] AIR rules ontology: `http://dig.csail.mit.edu/TAMI/2007/amord/air`

[12] AIR justifications ontology: `http://dig.csail.mit.edu/2009/AIR/airjustification.n3`

3.2.7. Semantic applications

Aside from the various representation mechanisms described earlier that support provenance encoding and personalized content, there have been many semantic applications (e.g., [49,66]) enabled by these representations that are explainable. We briefly describe two of these efforts.

In our automatic breast cancer characterization effort [66], we developed a visual interface to assist physicians in their diagnosis process by providing justifications of the treatment rules that resulted in a stage change of a patient between changing guideline editions. We considered the 7th and 8th edition of the American Joint Committee on Cancer (AJCC) cancer staging guidelines [24]. Our system reasoned using a knowledge base of encoded cancer staging rules, and inferred the stage of the patient based on their metastasis parameters and biomarkers. Our system could automatically determine the staging, explain how the stage was derived, and explain any restaging that happened. In another effort, McCusker et al. developed a framework [49] that encoded semantic connections between drugs, proteins, and diseases and allowed users to look for potential repurposing of drugs. A novel aspect of this system was that the interface allowed the user to explore why a drug may be used to target a particular disease, thus having a potential causal explanation as opposed to many other drug repurposing efforts that focused only on correlations. The system also included weights on all of the links in the graph so that users could get a sense of how strongly the evidence supports a relationship.

The semantic applications we reviewed primarily utilize scientific evidence to present factual content, discover new content, and automate human-intensive tasks. In Section 3.3, we review explanation modeling frameworks, such as, the Inference Web [53], which also have semantic representations but are used in more typical cognitive assistant settings.

3.2.8. Summary

The Semantic Web efforts we described address various components of explainability. Although, even these interpretable systems, powered by KGs and ontologies, do not entirely address all aspects of explainability that we detail in Section 2. However, we believe that semantic representations for explainability can evolve from the existing semantic representations for provenance, accountability and context. Hence, we believe that the strengths of the Semantic Web, coupled with ML methods, will be a significant contributor to hybrid explainable AI systems.

3.3. Cognitive Assistants

Cognitive Assistants are systems that are used to "augment human intelligence" [21] and aid humans in decision-making and problem-solving. These assistants have grown from their former role of professional assistants, educating users in a particular domain, to being widely accessible as personal assistants, aiding users in their everyday tasks. These assistants function in a tight coupling with the user and, hence, their design, knowledge bases, and interactions are driven by users' cognitive capabilities and needs. Further, these assistants play various roles from fostering positive behavior change, to training people with the necessary problem-solving skills in a domain, to providing tailored in-

[13] Image available at: http://dig.csail.mit.edu/TAMI/2008/JustificationUI/howto.html

formation based on an understanding of user context [19]. As the proliferation of general purpose, conversational cognitive assistants grows, it will become increasingly important that they include a representation of the user's goals, and "theory-of-mind" elements that support effective communication and collaboration [22].

3.3.1. DARPA PAL program

An ambitious and multi-university program, the Defense Advanced Research Projects Agency (DARPA) program, Personal Assistant that Learns (PAL),[14] gave rise to the Cognitive Assistant that Learns and Organizes (CALO) system. CALO was a large effort including over 20 collaboration organizations aimed at building a cognitive agent that can assist in a wide range of day-to-day office-related tasks, including sending out emails, memos, maintaining a to-do list [14], etc. Henceforth, several projects leveraged the CALO work, the most famous is Apple's personal assistant Siri. In our review, we will cover some of the seminal explainable cognitive assistants [53,54] and user studies [25] that resulted from or were refined within the CALO project, that are explainable in their own right.

Inference Web was one of the early modular explanation frameworks, and it built upon the strengths from the Semantic Web [32], Description Logics [4], and expert systems communities, to generate explanations for distributed, web-based systems that were interacting with users. The framework provided explanations that contained the provenance of the information (both implicit and explicit), and the proof for inference traces to novice users and agents alike. Additionally, the framework could abstract explanations to suit users' understanding and to avoid lengthy proofs that would overwhelm the users (similar to the breadcrumbs features provided by the CLASSIC system (Section 3.1.2)). Besides the ability to abstract explanations, the framework was also capable of providing explanations in different formats and even had a built-in explanation dialogue that would display questions and answers. Users could then interact with the answers and pose follow-up questions. The framework achieved its explanation capabilities via a modular architecture consisting of an IWBase, a data repository of the metainformation about the information used by the framework; an IWAbstractor, abstractor component that converted lengthy Proof Markup Language (PML) [15] proofs to PML explanations; an IWExplainer, an explanation dialogue component that would generate explanations for users; and an IWBrowser, a browser for displaying the explanations. While the Inference Web framework did not include a context-specific component, it provided some context modeling options and was capable of providing a wide range of customized explanation capabilities that included direct support for encoding trust and user models.

McGuinness et al. [54] expanded on their earlier Inference Web [53] framework, and developed an Integrated Cognitive Explanation Environment (ICEE) that generated explanations for task reasoning. ICEE served as an explanation component on the CALO system, in which multiple reasoning techniques, including task processing, numerous learning components, along with statistical and deductive methods, all worked together. Since CALO served as a cognitive assistant in the workplace, the tasks involved processing workplace automation activities, such as requesting quotes from different sources (e.g., *GetQuotes* was one of the sub-tasks [54]). Additionally, the reasoning techniques used in CALO used multiple knowledge sources to generate conclusions that needed to

[14] PAL: https://www.darpa.mil/about-us/timeline/personalized-assistant-that-learns

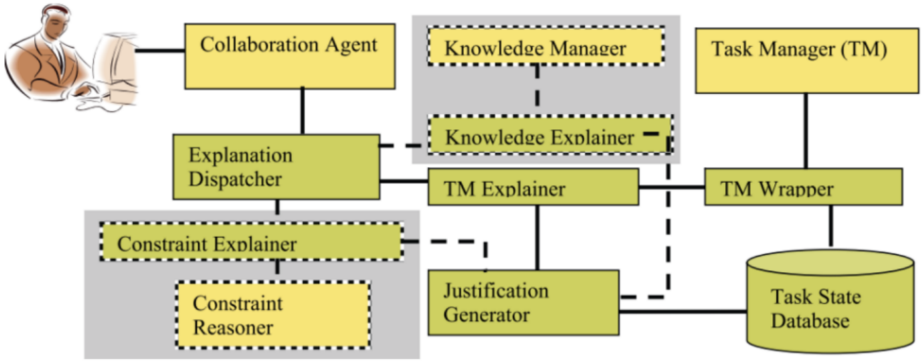

Figure 2. An activity flow diagram of the Integrated Cognitive Explanation Environment (ICEE) that was utilized to explain task-processing systems in the CALO project taken from McGuinness et al. [54].

be explained. The ICEE explanation architecture (shown in Figure 2) consisted of several components critical to generating explanations: an *explanation dispatcher* that interpreted a user's explanation request and invoked different reasoning components based on the type of explanation request, a *task manager explainer* that further invoked task manager wrappers to gather task execution information, a *task state database* that maintained the execution traces and states of the tasks, and a *justification generator* that created explanations from the task execution processing information.

The authors conducted a user study aimed at understanding the types of questions that users wanted answered. These explanation request types included questions about the motivation of a task, status, execution history, forward-looking execution plans, task ordering, or explicit questions about time [54]. The classifications of these explanation requests into different request types helped invoke appropriate explanation strategies. Additionally, the system hosted introspective predicates were used to identify the types of information to be included in explanations based on the request's intent. Broadly, the introspective predicates were grouped into *basic procedure information*, metadata about task definitions; *execution information*, details about the task execution; and, *projection information*, information about future task processing. The ICEE framework provides an example of many of the components needed in explainable hybrid AI systems and demonstrated how they can be used to provide user-customized explanations.

Another noteworthy effort from CALO was Glass, et al. [25]'s user study that assessed the trust and understandability aspects of adaptive systems. They used the CALO system as an adaptive system use case in their study. Their findings grouped users' concerns into eight themes: 1). *High-level usability of complex prototypes*, 2). *being ignored*, 3). *context-sensitive questions*, 4). *granularity of feedback*, 5). *transparency*, 6). *provenance*, 7). *managing expectations*, and 8). *autonomy and verification*. While there were some system-related concerns that could be addressed via system improvements (high-level usability, verification), there were also other concerns, such as, provenance, the granularity of feedback, the transparency targeting the users' perception of trust in the agent. They found that the trust level of most users in the system increased significantly with the inclusion of provenance and context-sensitive aspects. Therefore, this study concluded that users who work with cognitive agents would like an interactive dialogue and personalized experience and would prefer provenance information to under-

stand the working of these complex systems, to some degree. The themes identified in this paper remain desired features for our complex, hybrid systems of today that use both statistical ML and reasoning techniques.

3.3.2. Intelligent tutors

Intelligent tutoring is a sub-domain of cognitive assistants, where adaptive task-oriented systems are utilized for training humans in a particular domain. Hence, intelligent tutors need to appeal to the human cognition and understand and evolve their learning capabilities and grasp of the domain. In a seminal work, VanLeHan [75] noted that there are two loops to human tutoring, an inner and outer loop. He noted that the inner loop worked in tandem with the human, helping them at each step, assessing their competence, and updating the student model, while the outer loop identified a new task to execute based on the student's assessment. Enhancements have been proposed to VanLeHan's inner and outer loop proposition, one of which is a behavior graph [2] that kept track of the possible problem-solving strategies that students can adopt. The edges in a behavior graph represented the different ways in which students could solve problems, and the nodes represented the acceptable states. In general, intelligent tutors host an inherent, domain-specific knowledge component that is used to undertake tasks.

A use case on explainable, intelligent tutors was explored in a military setting by a Disciple-LTA [72] system. They used an iterative problem-solving approach in intelligence tasks to assist analysts. These tasks were broken down into executable steps to which evidence could be associated to find solutions (also termed as "task-reduction"). The solutions were then combined at the task level, or "solution-composition," to produce conclusions. A sample conclusion from this system was "There is strong evidence that Location-A is a training base for terrorist operations." [72] The Disciple-LTA architecture consisted of different reasoning agents: learners, tutors, and problem-solvers, all of which read from and wrote into the knowledge base of an ontology and its rules.

3.3.3. Summary

The cognitive assistant literature is vast and continues to grow with the emergence of personal assistants, such as, Apple's Siri, Amazon's Alexa, etc. In our review, we have covered explanation facilities in DARPA's CALO project [25,53,54], and have also briefly discussed Intelligent Tutors [2,72,75]. While the focus of explanations in the CALO cognitive assistant was on explaining task-based workflows, the underlying system contained a set of hybrid deductive reasoners coupled with numerous learning components, and thus is representative of today's hybrid learning systems. User requirements were utilized to design explanation strategies and determine the execution of the next task, dictated by user feedback. Cognitive assistants have begun to focus on the end-user, and are supporting facilities to account for user perspective, to some extent, unlike expert systems (Section 3.1) that focused primarily on generating explanations of inference traces.

3.4. Explainability in Machine Learning (ML)

ML algorithms have been rapidly advancing, proliferating in various domains, even high-precision domains, such as, healthcare and finance. However, these algorithms, are typically more opaque than previous expert systems (Section 3.1), semantic applications

(Section 3.2), and cognitive assistants (Section 3.3). Hence, the ML domain faces large challenges in addressing the trustworthiness, transparency, and intelligibility[15] of their models. Additionally, even within the ML domain, there has been a shift from the dependence on simpler linear algorithms that were less complex, to non-linear, "black-box" models, such as, deep learning [44]. While ML algorithms are often achieving high accuracy, they are typically unable to explain why they arrived at a classification or score (view the tradeoff in Figure 3). However, there have been techniques to circumvent these issues, such as, providing confidence scores for the results of models to induce trust (post-hoc interpretations [3,57]), attaching semantic information to results [5], presenting contrastive or counterfactual explanations to provide intuition for the model's functioning [74,76], etc. Formally, the interpretability techniques for ML models can be grouped into two categories [57], one class aimed at *post-hoc interpretations* that contain explanations about the results to provide perspective on the model's functioning, and the other aimed at improving *transparency* to offer an intuition for the model's functioning. We want to clarify that although ML models might not be considered traditional knowledge-enabled system candidates, we have included them in our review due to the emergence of hybrid systems composed of ML models and semantic methods. We believe that a review of explainability approaches in the ML domain will be fruitful for introducing explanation components into these hybrid systems.

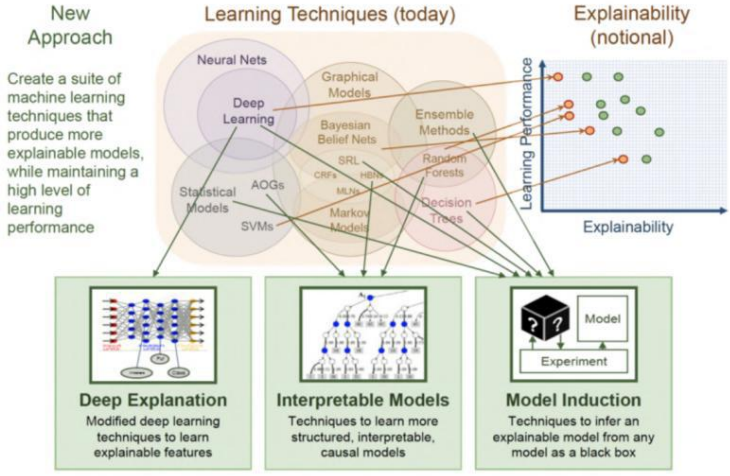

Figure 3. A high-level overview of the ML models' classes and the explanation techniques being developed as part of DARPA's eXplainable AI program. It is interesting to note the accuracy-explainability tradeoff depicted in the graph on the right, which shows that within the ML domain, simpler models which are oftentimes less accurate are often more explainable [Image taken from Gunning [27]].

[15] Our definition of intelligibility is very similar to the description proposed by Lipton [46] and Lou et al. [47], in that intelligible models are interpretable wherein the contribution of model features to a decision can be deciphered.

3.4.1. DARPA XAI Program

DARPA's eXplainable AI (XAI) program[16] focuses on building explainable models that achieve high accuracy and on methods to enable human users to trust and understand these models. We will discuss selected XAI efforts mentioned in the DARPA XAI reports [27,28] that have a knowledge explainability component to them.

Bau et. al. [5], have developed a *network dissection* technique to align the intermediate layer results of convolutional neural networks (CNN) with semantic concepts. They make two contributions, a *network dissection* technique to identify what the network is learning at each step by comparing it to semantic concepts, and the construction of *disentangled representation* to align encodings between the network's output and a semantic concept. The disentangled representations were designed to provide a notion of the "human perception of what it means for a concept to be mixed up" [5]. Further, the authors also assembled a new dataset, the Broadly and Densely Labeled Dataset (Broden) [78] of objects, that contained low-level compositions of objects used as semantic concepts. This work addresses the deep explanation component of Figure 3, wherein feature modifications are being made to make deep learning algorithms interpretable. Similarly, as part of the same program, a team of Charles River Analytica (CRA) researchers developed a technique to learn the causal nature of CNN activations [30]. In this work, Harradon et al. [30] construct a causal graph in-line with Judea Pearl's do-calculus [62] method. They ground the network activations in a $P(O,P,C)$ graph, where C represents concepts of network representations that humans can identify, P is the input, and O is the output. However, unlike the network dissection paper [5], the causal graph is learned via an unsupervised autoencoder method. Hence, it might be challenging to trust the causal graphs that are learned.

Since the XAI program by DARPA is an ongoing initiative, some of the work mentioned in the slideware[17] remains unpublished. However, we briefly summarize some of these unpublished methods that we believe are relevant to our explainability review. In the Common Ground Learning and Explanation (COGLE) project,[18] being led by PARC, a system is being built to explain to humans the workings of an autonomous Unmanned Aircraft System (UAS) testbed. The COGLE system explains the workings of the UAS reinforcement learning decision-making algorithm to users, conveys an understanding of the system's future behavior, and uses a common ground vocabulary to present these explanations. The common ground vocabulary is generated by including both human understandable and machine-understandable terminology, hence, hoping to ensure a dialogue between ML algorithms and humans. The common ground idea corroborates a requirement put forth by Doshi-Velez et al., [18] that "to build AI systems that can provide explanation in terms of human-interpretable terms, we must both list those terms and allow the AI system access to examples to learn them." In another effort, researchers at Rutgers have proposed a technique to choose optimal examples to explain a model's decision via Bayesian Teaching [77]. The explanation by the Bayesian Teaching method is an explanation by examples technique, wherein model-agnostic probabilistic methods

[16] DARPA XAI program website: https://www.darpa.mil/program/explainable-artificial-intelligence

[17] DARPA explainable AI slideware: https://www.darpa.mil/attachments/explainableAIProgramUpdate.pdf

[18] COGLE: https://www.parc.com/blog/explainable-ai-an-overview-of-parcs-cogle-project-with-darpa/

are used to identify the most probable data points that lead to a conclusion. The hypothesis that Yang et al. [77] present is that the data is most representative of the algorithm's conclusions, and humans tend to understand more intuitively through examples.

In summary, the DARPA XAI program (of which the report is a by-product [27]) is largely focused on improving explainability of deep learning models through local interpretation methods, or "knowing the reasons for specific decisions" [17] and post-hoc interpretations. These focus points, to some extent, address the trustworthiness and intelligibility aspects of the explainability of ML models.

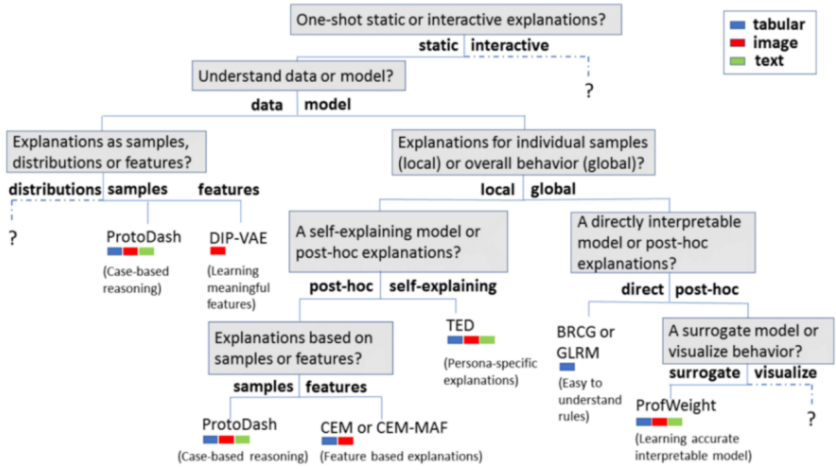

Figure 4. A decision-tree like visual overview of the taxonomy of explanations which encodes different factors ML models need to consider while designing explainable models. [Image taken from Arya et al. [3]]

3.4.2. Taxonomies in explainable ML

Besides the DARPA XAI program, there have been other recent efforts in the ML domain to support the explainability of ML models. A team of researchers from IBM Research have built the AI Fairness 360 [6] and AI Explainability 360 [3] toolkits to identify bias in datasets and ML algorithms, and to describe the explainability of ML models, respectively. In their AI fairness 360 toolkit, Bellamy et al. [6] define metrics to identify bias in three stages of the dataset, the algorithm, and the predictions of the algorithm, in their goal to improve fairness in the entire ML workflow. While, in Section 1, we noted that we do not account for fairness in explainability, we acknowledge that the exposition of the fairness of the algorithm and data could increase trust in the model. Furthermore, in the AI explainability 360 effort, Arya et al. [3] designed a taxonomy resource to provide a structure of the explanation space to benefit algorithm designers who are looking to include necessary components in explanations. More specifically, their taxonomy (Figure 4) helps in identifying methods to introduce local (explanations of portions of the model that lead to a conclusion), global (an explanation of the entire model), and post-hoc interpretations (explaining the results) of models via careful inclusion of features and mechanisms during the design of ML models. However, their taxonomy focused more

on model interpretability and features of the model, rather than on intended use of the model by users.

Researchers at MIT conducted a literature review of published explainable AI papers and cataloged the explanation methods used by ML algorithms into a taxonomy [23]. Their taxonomy grouped papers into three categories, methods that emulate the processing of the data, explanations of representations (such as, the network dissection technique [5]), and explanation-producing networks. Their hope was for future methods to use the taxonomy as a reference to build explainable models. Additionally, they note that certain ML methods, such as, decision trees are more interpretable than black-box models and hence are being utilized as proxies [79] to explain the conclusions of these black-box models. Similarly, Gilpin et al. [23] proposed the Local Interpretable Model-agnostic Explanations (LIME) framework [63], that can be utilized to generate linear models on perturbations of the black-box model input to get a sense of the functioning of the black-box models.

3.4.3. Summary

From a review of the ML domain, we can infer that the explainability techniques being developed are mainly tackling challenges of model interpretability and generating post-hoc interpretations of the model's conclusions or input data. While these two broad categories might seem insufficient, the breadth of innovative approaches [3,5,76,77] being developed are promising and can help in building interpretable, and hybrid models, aided by explainable models (e.g., KGs, causal methods). In summary, what makes the models that we describe in this section candidates for explainable knowledge-enabled systems is that they utilize knowledge to provide an intuition for the functioning of unintelligible models [5], or to build a vocabulary (COGLE: [19]) to explain conclusions/inputs/workings of the algorithms. Additionally, prior knowledge of the requirements of explanations are being encoded as taxonomies [3,23] to serve as checks for future explainable models, and knowledge of existing linear models [63,79] are being leveraged to enhance the explanation capabilities of ML models. Orthogonally, the interpretability research in the ML domain is helping researchers understand that humans prefer richer, social, contrasting, and selective explanations [57].

4. Conclusion

We presented foundational approaches to explainable, knowledge-enabled systems, and identified themes for explainability within these approaches. We presented our definition of "explainable knowledge-enabled systems" to cover a broad range of past and present AI systems including expert systems, Semantic Web, cognitive assistants, and ML domains. Additionally, we believe that, with the increasing focus on explainable AI, we are at the cusp of a new era of AI where explainability plays a pivotal role in the adoption of AI systems. We provided synthesized, refined definitions of knowledge-enabled systems from a user perspective and included properties that are desirable for when a system needs to generate *provenance-aware, personalized, and context-aware* explanations.

[19] COGLE: https://www.parc.com/blog/explainable-ai-an-overview-of-parcs-cogle-project-with-darpa/

We reminded our readers that different AI domains and varying methodologies are differently suited for various aspects of explanations. The next-generation hybrid AI systems would benefit from these identified strengths, utilizing a potentially, carefully chosen combination of these techniques to provide more complete, satisfying explanations. For instance, we identified that trace-based explanation facilities are well-explored in expert systems, provenance encoding in the Semantic Web domain is capable of representing different granularities of evidence, the modular, task-based explanation facilities of cognitive assistants can generate atomic explanation components, and that interpretability efforts in the ML domain are giving rise to taxonomical checks for explainable AI models that can be adapted to other AI fields. However, we noted that these AI systems do not fully account for aspects such as user context and causality and are only capable of generating explanations belonging to a restricted set of explanation types. To address these issues, we present directions for research and describe different explanation types in a later chapter, "Directions for Explainable Knowledge-enabled Systems," that might play a key role in furthering explainable AI.

In conclusion, we believe that with the increased adoption of AI systems, there is an increased need for systems to be interpretable, adaptive, interactive, and, most importantly, able to generate explanations that not only provide an overview of the AI system, but serve as a means to educate users and help in their future explorations. To address these lofty goals of explainability, we believe that we need to learn from strengths of past foundational approaches and adapt/expand on them in the user-centric needs of the current AI landscape to build hybrid AI systems that are interpretable, knowledge-enabled, adaptive, and context and provenance-aware.

5. Acknowledgments

This work is partially supported by IBM Research AI through the AI Horizons Network. We thank our colleagues from IBM Research, Amar Das, Morgan Foreman and Ching-Hua Chen, and from RPI, James P. McCusker, and Rebecca Cowan, who greatly assisted the research and document preparation.

References

[1] Nkcheniyere N. Agu, Neha Keshan, Shruthi Chari, Oshani Seneviratne, James P. McCusker, Amar Das, and Deborah L. McGuinness. G-prov: Provenance management for clinical practice guidelines. In *Proc. of the Semantic Web solutions for large-scale biomedical data analytics Workshop*, page to appear. CEUR, 2019.

[2] Vincent Aleven, Bruce M Mclaren, Jonathan Sewall, and Kenneth R Koedinger. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *Int. J. of Artificial Intelligence in Education*, 19(2):105–154, 2009.

[3] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[4] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.

[5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 6541–6549, 2017.

[6] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[7] Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3logic: A logical framework for the world wide web. *Theory and Practice of Logic Programming*, 8(3):249–269, 2008.

[8] Tim Berners-Lee, Dimitri Dimitroyannis, A John Mallinckrodt, and Susan McKay. World wide web. *Computers in Physics*, 8(3):298–299, 1994.

[9] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[10] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, page 1, 2017.

[11] Alexander Borgida, Ronald J Brachman, Deborah L McGuinness, and Lori Alperin Resnick. Classic: A structural data model for objects. In *ACM Sigmod record*, volume 18, pages 58–67. ACM, 1989.

[12] Ronald J Brachman and James G Schmolze. An overview of the kl-one knowledge representation system. In *Readings in artificial intelligence and databases*, pages 207–230. Elsevier, 1989.

[13] William J Clancey and Reed Letsinger. *NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching*. Dept. of Computer Sci., Stanford University Stanford, 1982.

[14] Kenneth Conley and James Carpenter. Towel: Towards an intelligent to-do list. In *AAAI Spring Symp.: Interaction Challenges for Intelligent Assistants*, 2007.

[15] Paulo Pinheiro Da Silva, Deborah L McGuinness, and Richard Fikes. A proof markup language for semantic web services. *Information Systems*, 31(4-5):381–395, 2006.

[16] Jasbir S Dhaliwal and Izak Benbasat. The use and effects of knowledge-based system explanations: theoretical foundations and a framework for empirical evaluation. *Information systems research*, 7(3):342–362, 1996.

[17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[18] Finale Doshi-Velez, Mason Kortz, Ryan Budish, Chris Bavitz, Sam Gershman, David O'Brien, Stuart Schieber, James Waldo, David Weinberger, and Alexandra Wood. Accountability of AI under the law: The role of explanation. *arXiv preprint arXiv:1711.01134*, 2017.

[19] Maria R Ebling. Can cognitive assistants disappear? *IEEE Pervasive Computing*, 15(3):4–6, 2016.

[20] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48, 2016.

[21] Douglas C Engelbart. Toward augmenting the human intellect and boosting our collective iq. *Communications of the ACM*, 38(8):30–33, 1995.

[22] Robert G Farrell, Jonathan Lenchner, Jeffrey O Kephjart, Alan M Webb, Michael J Muller, Thomas D Erikson, David O Melville, Rachel KE Bellamy, Daniel M Gruen, Jonathan H Connell, et al. Symbiotic cognitive computing. *AI Magazine*, 37(3):81–93, 2016.

[23] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*, 2018.

[24] Armando E Giuliano, James L Connolly, Stephen B Edge, Elizabeth A Mittendorf, Hope S Rugo, Lawrence J Solin, Donald L Weaver, David J Winchester, and Gabriel N Hortobagyi. Breast cancer—major changes in the American Joint Committee on Cancer eighth edition cancer staging manual. *CA: A Cancer J. for Clinicians*, 67(4):290–303, 2017.

[25] Alyssa Glass, Deborah L McGuinness, and Michael Wolverton. Toward establishing trust in adaptive agents. In *Proc. of the 13th Int. Conf. on Intelligent user interfaces*, pages 227–236. ACM, 2008.

[26] Paul Groth, Andrew Gibson, and Jan Velterop. The anatomy of a nanopublication. *Information Services & Use*, 30(1-2):51–56, 2010.

[27] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

[28] David Gunning and David W Aha. Darpa's explainable artificial intelligence program. *AI Magazine*, 40(2):44–58, 2019.

[29] Amelie Gyrard, Manas Gaur, Saeedeh Shekarpour, Krishnaprasad Thirunarayan, and Amit Sheth. Personalized health knowledge graph. In *Int. Semantic Web Conf. (ISWC) 2018 Contextualized Knowledge Graph Workshop*, 2018.

[30] Michael Harradon, Jeff Druce, and Brian Ruttenberg. Causal learning and explanation of deep neural networks via autoencoded activations. *arXiv preprint arXiv:1802.00541*, 2018.
[31] Rakebul Hasan and Fabien Gandon. Explanation in the semantic web: a survey of the state of the art, 2012.
[32] James Hendler and Eric Miller. Integrating applications on the semantic web. *J. of the Institute of Electrical Engineers of Japan*, Vol 122(10):676–680, 2002.
[33] Sven Hertling and Heiko Paulheim. Webisalod: providing hypernymy relations extracted from the web as linked open data. In *Int. Semantic Web Conf.*, pages 111–119. Springer, 2017.
[34] Aidan Hogan, Andreas Harth, and Axel Polleres. Scalable authoritative owl reasoning for the web. *Int. J. on Semantic Web and Information Systems (IJSWIS)*, 5(2), 2009.
[35] James Hollan, Edwin Hutchins, and David Kirsh. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2):174–196, 2000.
[36] Andreas Holzinger, Chris Biemann, Constantinos S Pattichis, and Douglas B Kell. What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017.
[37] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in owl. In *Int. semantic web Conf.*, pages 323–338. Springer, 2008.
[38] Edwin Hutchins. The technology of team navigation, intellectual teamwork: social and technological foundations of cooperative work, l, 1990.
[39] Lalana Kagal. Accountability In RDF (AIR) Web Rule Language, 2009.
[40] Johan de Kleer, Jon Doyle, Guy L Steele Jr, and Gerald Jay Sussman. AMORD explicit control of reasoning. *ACM SIGPLAN Notices*, 12(8):116–125, 1977.
[41] Tobias Kuhn, Paolo Emilio Barbano, Mate Levente Nagy, and Michael Krauthammer. Broadening the scope of nanopublications. In *Extended Semantic Web Conf.* Springer, 2013.
[42] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. *W3C recommendation*, 2013.
[43] Freddy Lecue. On the role of knowledge graphs in explainable ai. *Semantic Web J. (Forthcoming)*, 2019.
[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[45] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 2119–2128. ACM, 2009.
[46] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.
[47] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proc. of the 18th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 150–158. ACM, 2012.
[48] Rosa Maimone, Marco Guerini, Mauro Dragoni, Tania Bailoni, and Claudio Eccher. Perkapp: A general purpose persuasion architecture for healthy lifestyles. *J. of biomedical informatics*, 82:70–87, 2018.
[49] James P McCusker, Michel Dumontier, Rui Yan, Sylvia He, Jonathan S Dordick, and Deborah L McGuinness. Finding melanoma drugs through a probabilistic knowledge graph. *PeerJ Computer Sci.*, 3:e106, 2017.
[50] James P McCusker, Sabbir M Rashid, Zhicheng Liang, Yue Liu, Katherine Chastain, Paulo Pinheiro, Jeanette A Stingone, and Deborah L McGuinness. Broad, interdisciplinary science in tela: An exposure and child health ontology. In *Proc. of the 2017 ACM on Web Sci. Conf.*, pages 349–357. ACM, 2017.
[51] Deborah L McGuinness. *Explaining reasoning in description logics*. PhD thesis, Rutgers University, 1996.
[52] Deborah L McGuinness and Alexander Borgida. Explaining subsumption in description logics. In *IJCAI (1)*, pages 816–821, 1995.
[53] Deborah L McGuinness and Paulo Pinheiro Da Silva. Explaining answers from the semantic web: The inference web approach. *Web Semantics: Sci., Services and Agents on the World Wide Web*, 1(4):397–413, 2004.
[54] Deborah L McGuinness, Alyssa Glass, Michael Wolverton, and Paulo Pinheiro Da Silva. Explaining task processing in cognitive assistants that learn. In *AAAI Spring Symp.: Interaction Challenges for Intelligent Assistants*, pages 80–87, 2007.
[55] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[56] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.
[57] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. In *Proc. of the Conf. on fairness, accountability, and transparency*, pages 279–288. ACM, 2019.
[58] Donald A Norman. *The psychology of everyday things.* Basic books, 1988.
[59] Ingrid Nunes and Dietmar Jannach. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27(3-5):393–444, 2017.
[60] Peter F Patel-Schneider, Deborah L McGuinness, Ronald J Brachman, and Lori Alperin Resnick. The classic knowledge representation system: Guiding principles and implementation rationale. *ACM SIGART Bulletin*, 2(3):108–113, 1991.
[61] Judea Pearl. *Causality*. Cambridge university press, 2009.
[62] Judea Pearl. Theoretical impediments to machine learning, 2017.
[63] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.
[64] Mireia Ribera and Àgata Lapedriza. Can we do better explanations? a proposal of user-centered explainable ai. In *IUI Workshops*, 2019.
[65] Elaine Rich. Stereotypes and user modeling. In *User models in dialog systems*, pages 35–51. Springer, 1989.
[66] Oshani Seneviratne, Sabbir M Rashid, Shruthi Chari, James P McCusker, Kristin P Bennett, James A Hendler, and Deborah L McGuinness. Knowledge integration for disease characterization: A breast cancer example. In *Int. Semantic Web Conf.*, pages 223–238, San Francisco, USA, 2018. Springer.
[67] Edward Shortliffe. *Computer-based medical consultations: MYCIN*. Elsevier, 2012.
[68] Edward Hance Shortliffe. Mycin: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Technical report, Dept. of Computer Sci., Stanford University Stanford, 1974.
[69] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proc. of the 13th Int. Conf. on World Wide Web*, pages 675–684. ACM, 2004.
[70] William Swartout, Cecile Paris, and Johanna Moore. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, 6(3):58–64, 1991.
[71] William R Swartout and Johanna D Moore. Explanation in second generation expert systems. In *Second generation expert systems*, pages 543–585. Springer, 1993.
[72] Gheorghe Tecuci, Mihai Boicu, Cindy Ayers, and David Cammons. Personal cognitive assistants for military intelligence analysis: Mixed-initiative learning, tutoring, and problem solving. In *First Int. Conf. on Intelligence Analysis*, pages 2–6. Citeseer, 2005.
[73] Joshua Valdez, Matthew Kim, Michael Rueschman, Vimig Socrates, Susan Redline, and Satya S Sahoo. Provcare semantic provenance knowledgebase: evaluating scientific reproducibility of research studies. In *AMIA Annual Symp. Proc.*, volume 2017, page 1705. American Medical Informatics Association, 2017.
[74] Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470*, 2018.
[75] Kurt Vanlehn. The behavior of tutoring systems. *Int. J.of artificial intelligence in education*, 16(3):227–265, 2006.
[76] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gpdr. *Harv. JL & Tech.*, 31:841, 2017.
[77] Scott Cheng-Hsin Yang and Patrick Shafto. Explainable artificial intelligence via bayesian teaching. In *NIPS 2017 workshop on Teaching Machines, Robots, and Humans*, 2017.
[78] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.
[79] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *Int. Conf. on Discovery Sci.*, pages 457–473. Springer, 2016.

Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges 49
I. Tiddi et al. (Eds.)
IOS Press, 2020

doi:10.3233/SSW200011

Knowledge Graph Embeddings and Explainable AI

Federico BIANCHI [a], Gaetano ROSSIELLO [b], Luca COSTABELLO [c], Matteo PALMONARI [d], and Pasquale MINERVINI [e]
[a] *Bocconi University*
[b] *IBM Research AI*
[c] *Accenture Labs*
[d] *University of Milan-Bicocca*
[e] *University College London*

Abstract. Knowledge graph embeddings are now a widely adopted approach to knowledge representation in which entities and relationships are embedded in vector spaces. In this chapter, we introduce the reader to the concept of knowledge graph embeddings by explaining what they are, how they can be generated and how they can be evaluated. We summarize the state-of-the-art in this field by describing the approaches that have been introduced to represent knowledge in the vector space. In relation to knowledge representation, we consider the problem of explainability, and discuss models and methods for explaining predictions obtained via knowledge graph embeddings.

Keywords. Knowledge Graphs, Knowledge Graph Embeddings, Knowledge Representation, eXplainable AI

1. Introduction

A knowledge graph [39] (KG) is an abstraction used in knowledge representation to encode knowledge in one or more domains by representing entities like `New York City` and `United States` (i.e., nodes) and binary relationships that connect these entities; for example, `New York City` and `United States` are connected by the relationship `country`, i.e., `New York City` has `United States` as a `country`. Most of KGs also contains relationships that connect entities with *literals*, i.e., values from known data structures such as strings, numbers, dates, and so on; for example a relationship `settled` that connects `New York City` and the integer `1624` describe a property of the entity `New York City`. More in general, we can view a KG under a dual perspective: as a *directed labeled multi-graph*, where nodes represent entities or literals and labeled edges represent specific relationships between entities or between an entity and a literal, and as a set of *statements*, also referred to as *facts*, having the form of subject-predicate-object triples, e.g., (`New York City`, `country`, `United States`) and (`New York City`, `settled`, `1624`). In the following, we will use the notation (h, r, t) (head, relation, tail) to identify a statement in KG, as frequent in the literature about KG embeddings.

The entities described in KGs are commonly organized using a set of *types*, e.g., `City` and `Country`, also referred to as concepts, classes or data types (when referred

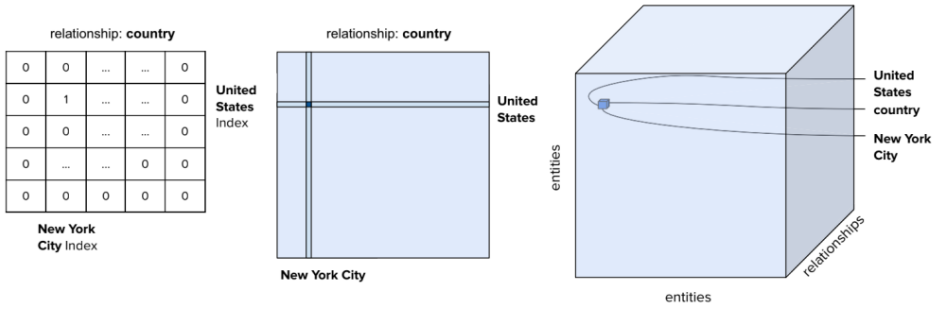

Figure 1. Binary adjacency representation of a KG.

to literals). For example, the statement (`New York City`, `type`, `City`) states that the entity `New York City` has type `City`. Indeed, this types are often defined in what is generally referred to as the *ontology* [21]. An ontology is a formal specification of the meaning of types and relationships expressed as a set of logical constraints and rules, which support automated reasoning. For example, DBpedia [3], a knowledge graph built upon information extracted from Wikipedia, describes more than 4 million entities and has 3 billion statements[1].

While KGs can be described using a graph, a nice and simple way to visualize a knowledge graph is considering it as a 3-order adjacency tensor (i.e., a 3-dimensional tensor describing the structure of the KG). Formally a 3-dimensional adjacency tensor is defined as $T \in \mathbb{R}^{N \times R \times N}$, where N is the number of entities and R is the number of relationships. Each dimension of the tensor corresponds to (`head`, `relation`, `tail`) respectively.

More formally, assume we have a KG $\mathscr{G} = \{(e_i, r_j, e_k)\} \subseteq \mathscr{E} \times \mathscr{R} \times \mathscr{E}$, where $\mathscr{E}$ and $\mathscr{R}$ denote the sets of entities and relations in the KG, respectively, with $|\mathscr{E}| = N$ and $|\mathscr{E}| = R$. The adjacency tensor $T \in \mathbb{R}^{N \times R \times N}$ is defined as follows:

$$T_{i,j,k} = \begin{cases} 1 & \text{if } (e_i, r_j, e_k) \in \mathscr{G}, \\ 0 & \text{otherwise.} \end{cases}$$

To visualize this, imagine a simple adjacency matrix that represents a single relation, such as the `country` relation: the two dimensions of the matrix correspond to the head entity and the tail entity. Each entity corresponds to an unique index: given a triple (`New York City`, `country`, `United States`), we have a 1 in the cell of the matrix corresponding to the intersection between the i-th row and the j-th column, where $i, j \in \mathbb{N}$ are the indices associated with `New York City` and `United States`, respectively. On the other hand, any cell in the adjacency matrix corresponding to triples not in the KG contains a 0. If we consider more than one relationship and we stack them together, we obtain a 3-dimensional tensor, generally referred to as the binary tensor representation of a KG. See Figure 1 for a simple visualization of this concept.

[1] https://wiki.dbpedia.org/about/facts-figures

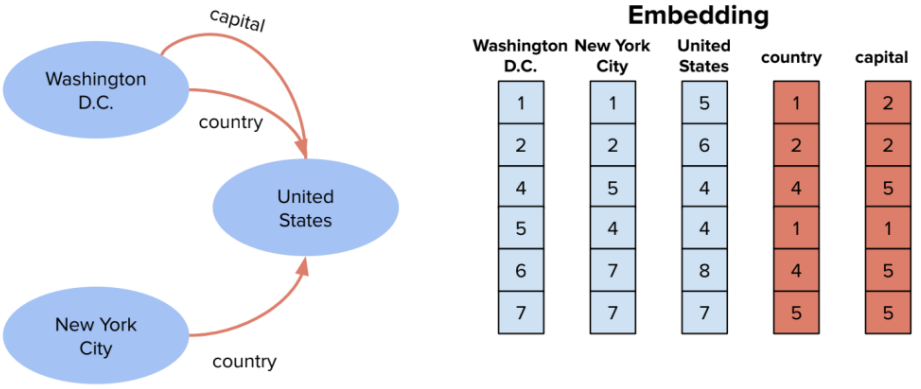

Figure 2. Starting from a knowledge graph, embedding methods generate representations of the elements of the knowledge graph that are embedded in a vector space. For example, these representations could be vectors. Vectors encode latent properties of the graph and for example similar entities tend to be described with similar vectors.

The term "knowledge graph embeddings" refers to the generation of vector representations of the elements that form a knowledge graph[2]. Essentially, what most methods do is to create a vector for each entity and each relation; these embeddings are generated in such a way to capture latent properties of the semantics in the knowledge graph: *similar* entities and *similar* relationships will be represented with *similar* vectors. Figure 2 provides an intuitive example of what a knowledge graph embedding method does. The tensor representation introduced above is frequently used in many KG embedding methods that learn embeddings by using dimensionality reduction techniques over the tensor.

The elements are generally represented in a vector space with low dimensionality (with values ranging from 100 dimensions to 1000 dimensions) and one key aspect is given by the notion of similarity: in a vector space similarity can be interpreted with the use of vector similarity measures (e.g., cosine similarity, in which two vectors are more similar if the angle between them is small).

An important task is to find ways to extend KGs adding new relationships between entities. This task is generally referred to as link prediction or knowledge graph completion. Adding new facts can be done with the use of logical inference. For example, from a triple (`Washington D.C.`, `capital`, `United States`) we can infer (`Washington D.C.`, `country`, `United States`). Inferring this last fact comes from background knowledge encoded in an axiom that specify that if a city is a capital of a country, it is also part of that country (e.g., as encoded by a first order logic rule such as $\forall X, Y : \text{capital}(X, Y) \Rightarrow \text{country}(X, Y)$). Unfortunately, many knowledge graphs have many observed facts and fewer axioms or rules [87].

KG embeddings can be used for link prediction, since they show interesting predictive abilities and are not directly constrained by logical rules. This property comes at the

[2]Note that knowledge graph embeddings are different from Graph Neural Networks (GNNs). KG embedding models are in general shallow and linear models and should be distinguished from GNNs [78], which are neural networks that take relational structures as inputs.

cost of not being directly interpretable (i.e., the vector representations now encode the latent meaning of the entity/relationship). The explainability of this prediction is often difficult because the result comes from the combination of latent factors that are embedded in a vector space and an evaluation of the inductive abilities of these methods is still an open problem [87].

Knowledge graph embeddings projected in the vector space tend to show interesting latent properties [61]; for example, *similar* entities tend to be close in the vector space. The value of similarity in the latent space is a function that depends on the way knowledge graph embeddings are generated. Similarity is also important under the point of view of explaining the meaning. For instance, we might not know the meaning of the entity `New York City`, but it can be inferred from its topic by looking at closest entities in the geometric space (i.e. `Washington D.C.` and `United States`).

The components of the vectors representing the entities and relations are not explainable themselves, and it can be hard to assign a natural language label that describes the meaning of that component. However, we can observe how different entities and relationships are related within the graph by analyzing its structure – which was also used to generate the vector-based representations. In addition, the training is driven by a similarity principle, which can be easily understood. For example, similar entities have similar embedding representations, and the same is true for similar relationships. Thus, while it is not possible to explain the exact difference between two vectors of two entities, we can refer to this similarity when using the vectors in more complex neural networks that use these vectors and the additional information to enrich the network capabilities.

Knowledge graph embeddings have been used in different contexts including recommendation [40,91,106], visual relationship detection [4] and knowledge base completion [11]. Moreover, knowledge graph embeddings can be used to integrate semantic knowledge inside deep neural networks, thus enriching the explainability of pure black-box neural networks [48,38], but they also come with some limitations.

In this chapter, we describe how to build embedding representations for knowledge graphs and how to evaluate them. We discuss related work of the field by mentioning the approaches that improved the state-of-the-art results. Then, we focus on knowledge graph embeddings to support explainability, i.e. how knowledge graph embeddings can be adopted to provide explanations by describing the relevant state-of-the-art approaches. Similarity comes has a key factor also in the context of explainability, in recommender systems for example, similarity is a key notion to express suggestions to users.

1.1. Overview of this Chapter

This chapter provides an overview of the field in which we describe how KG embeddings are generated and which are the most influential approaches in the filed up to date. Moreover, the chapter should also describe which are the possible usages for KG embeddings in the context of explainability. In the recent literature, many approaches for knowledge graph embeddings have been proposed; we summarize the most relevant models by focusing on the key ideas and their impact on the community.

In Section 2 we give a more detailed overview related to how a knowledge graph embedding method can be defined and trained. We will describe TransE [11], one of the most popular models, and then we will briefly explain how information that does not come from the knowledge graph can be used to extend the capabilities of the embedding

models. This will be a general introduction that should help the reader understand how the methods introduced in the other sections work.

In Section 3, we describe the approaches we have selected. We summarize what researchers have experimented within the field, giving to the reader the possibility of exploring different possible ways of generating knowledge graph embeddings. Note that it is difficult to describe which is the best model for a specific task because evaluation results are greatly influenced by hyper-parameters (see Section 3.5). Nevertheless, we think that most of the approaches have laid the basis for further development in the field and are thus worth describing. We then describe how knowledge graph embeddings are evaluated, showing that the main task is link prediction and that the datasets used have changed over the years. Link prediction is a task that requires high explainability, something that in the context of knowledge graph embeddings is often missing. In general, ComplEx [88] is often considered as one of the best performing models [4] and gives stable results in inductive reasoning tasks [87].

Then, in Section 4, we focus on explainability. Explainability is a difficult term to define [53]. Knowledge graph embeddings are not explainable by default, because they are sub-symbolic representations of entities in which latent factors are encoded. Knowledge graph embeddings can be used for link prediction, but the prediction is the result of the combination of latent factors that are not directly interpretable. However, there is recent literature that explores the usage of embeddings in the context of explainable and logical inferences.

We conclude this chapter in Section 5, where we summarize our main conclusions and we describe possible future directions for the field.

Additional Resources Several works that provide an overview of knowledge graph embeddings have been proposed in the literature. We point the reader to [28] that contains a nicely written survey of approaches that are meant to support the embedding of knowledge graph literals and to [92] for another overview on knowledge graph embeddings. As knowledge graph embeddings provide sub-symbolic representations of knowledge there is a recent increasing interest in finding ways to interpret how these representations interact [1]. Inductive capabilities of knowledge graph embeddings methods have been recently evaluated [87].

2. Knowledge Graph Embeddings

A Short Primer In this first part, we are going to define the general elements that characterize a knowledge graph embedding method. To better illustrate how knowledge graph embeddings are created we focus our explanation on one of the seminal approaches of the field, TransE [11]. We will introduce how TransE embeddings can be generated and how a method like TransE can be extended to consider information that is not included in the set of triples. While we will describe TransE-specific concepts, most of what it is explained in this section is still valid for other methods in the state of the art.

Nowadays, a plethora of approaches to generate embedded representations of KGs exists [11,67,96,52,88]. In 2011, RESCAL [67] was the first influential model to create embedded representations of entities and relationships from a KG by relying on a tensor factorization approach upon the 3-dimensional tensor generated by considering subject entity, predicate entity and object entity as the 3 dimensions of the tensor. There

are mainly three elements that are used to distinguish a method to generate KGs embedding: (i) the choice of the representations of entities and relationships, in general vector representations of real numbers are used [11,96], but there are methods that use matrices to represent relationships [67] and complex vectors to represent entities and relationships [88]; (ii) the so-called scoring function, which we will refer to as ϕ. This function is used to aggregate the information combing from a triple, and is generally referred to as the function that estimates the *likelihood* of the triple; lastly (iii) the loss function, which defines the objective being minimized during the training of the knowledge graph embedding model.

Changes in these three elements is what generally makes one model better than the other (although, see Section 3.5, where we explain the impact of different hyperparameters on the comparison). Scoring functions can be extended with many different information like, information coming from images [98] or numerical and relational features [26], in which the entity vector of a scoring function might be represented with the aggregation of image representations of that entity or textual content, an entity can be represented by aggregating the information contained inside its textual description. At the same time, loss functions can be extended considering different parameters, e.g., it is possible to extend a loss function by adding regularization. The interaction between the entity vectors and the relationship vectors is modulated by the score function. The score function computes a confidence value of the likelihood of a triple.

The learning process requires both positive and negative data in input and KGs contain only positive information. In KG embeddings the generation of negative is generally achieved generating *corrupted triples* i.e., triples that are false. For example, if in a knowledge graph we have the triple (`New York City`, `country`, `United States`), a simple corrupted triple is (`United States`, `country`, `New York City`). Note that despite these training procedures might have several limitations, different methods have been proposed to optimize the selection of good negative samples. One of the most advanced techniques is KBGAN [13] that proposes an adversarial method to generate effective negative training examples that can improve the representations of the knowledge graph embedding.

Making Knowledge Graph Embeddings TransE [11] uses k-dimensional vectors to represent both entities and relationships; the score function that the authors propose as the following form $d(\mathbf{h}+\mathbf{r},\mathbf{t})$, where the d function can be the L1 or the L2 norm. The driving idea of this score function is that the sum of the subject vector with the predicate vector should generate the vector representation of the object as output (i.e. $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$), in general the scoring function can be also defined as $d(\mathbf{h}+\mathbf{r},\mathbf{t}) = \|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$. The loss function defined to learn the representations is instead:

$$\mathscr{L} = \sum_{h,r,t \in S} \sum_{h',r,t' \in S'_{h,r,t}} [\gamma + d(\mathbf{h}+\mathbf{r},\mathbf{t}) - d(\mathbf{h}'+\mathbf{r},\mathbf{t}')]_+,$$

where $[x]_+$ is the positive part of x and γ is a margin hyper-parameter. And $S'_{h,r,t}$ is the set of corrupted triples. $d(\mathbf{h}+\mathbf{r},\mathbf{t})$ is the score of the true triple while $d(\mathbf{h}'+\mathbf{r},\mathbf{t}')$ is the score of the true triple. This loss function favors low values of $d(\mathbf{h}+\mathbf{r},\mathbf{t})$ with respect to the corrupted triples, in such a way that the function can be effectively minimized. It is possible to optimize the representation through the use of gradient-based techniques that are now common in machine learning. Figure 3 shows how TransE combine entities and

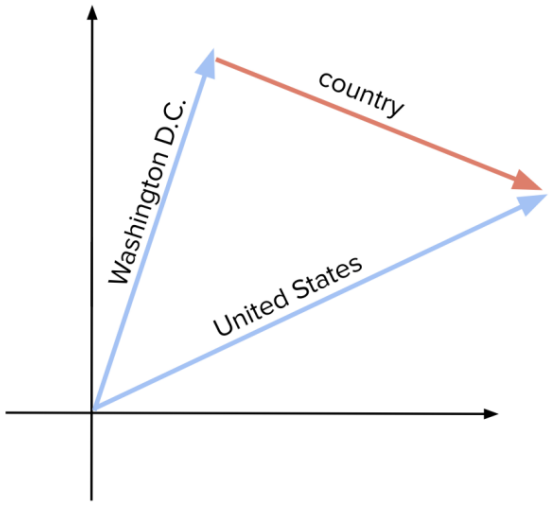

Figure 3. Example of how TransE represents and models the interactions between entities and relationships in vector space.

relationships in the scoring function. Through the training process, TransE learns vector representations of entities and relationships.

Augmenting Knowledge Graph Embeddings Knowledge graph embeddings can be generated by considering information that is not included in the graph itself. Different methods have been introduced to extend knowledge graph embeddings by adding novel information outside from the one provided by knowledge graph triples and we will give a more detailed overview in the next section, here we describe a method that extends TransE using textual information; adding elements to the score function allows us to include novel information inside our representations.

Description-Embodied Knowledge Representation Learning (DKRL) [100] jointly learns a structure-based representation h_s (as TransE) and a description-based representation t_d that can be used in an integrated scoring function, thus combining the relative information coming from both text and facts. To extend with additional information a model like TransE, the scoring function can be extended to optimize also other representations. For example, DKRL uses the following scoring function:

$$\|\mathbf{h}_s + \mathbf{r} - \mathbf{t}_s\| + \|\mathbf{h}_d + \mathbf{r} - \mathbf{t}_d\| + \|\mathbf{h}_s + \mathbf{r} - \mathbf{t}_d\| + \|\mathbf{h}_d + \mathbf{r} - \mathbf{t}_s\| .$$

Optimizing this joint score function allows us to combine the information coming from both text and triples. In detail, DKRL uses convolutional neural networks to generate description based representations for the entities. Different information can be used to extend the embedding such as images, logical rules, and textual information. In general, the process to introduce new information relies on the extension of the scoring function. Often adding more information allows us to extend the capabilities of the model. For example, the use of text-based representations allows us to generate vector representations of entities for which we have a description but that are not present in the KG.

Method	Scoring Function	Representation
RESCAL [67], 2011	$\mathbf{h}^\intercal \mathbf{W}_r \mathbf{t}$	$\mathbf{h},\mathbf{t} \in \mathbb{R}^d$, $\mathbf{W}_r \in \mathbb{R}^{d \times d}$
TransE [11], 2013	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^d$
DistMult [103], 2014	$\langle \mathbf{h},\mathbf{r},\mathbf{t} \rangle$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^d$
HolE [66], 2016	$\langle \mathbf{r}, \mathbf{h} \otimes \mathbf{t} \rangle$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^d$
ComplEx [88], 2016	$\mathrm{Re}(\langle \mathbf{h},\mathbf{r},\bar{\mathbf{t}} \rangle)$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{C}^d$
RotatE [82], 2019	$-\|\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|\|^2$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{C}^d$, $\|r_i\| = 1$

Table 1. A short list with knowledge graph embedding approaches with the respective scoring functions and the representation space used for entities and relationships. Lowercase elements are vectors, while uppercase elements are matrices, $\otimes$ is the circular correlation. $\bar{\mathbf{t}}$ defines the complex conjugate of an $\mathbf{t}$ and Re denotes the real part of a complex vector. We sampled these approaches by considering the novelty they introduced at the time they were presented. Score functions are based on those published in [82,6].

3. State-of-the-art Knowledge Graph Embeddings

In this section, we review some of the algorithms that have been introduced in the state of the art. Our main objective is to give the reader an overview of the research that has been done until now and which are the key points in the knowledge graph embedding field.

3.1. Structure-based Embeddings

Approaches that focus on the use of knowledge graph facts have also been called *fact alone* methods by other authors [92]. Table 1 shows the different scoring function that can be used to define different knowledge graph embeddings methods. The two main categories of approaches are the *translational models* and the *bilinear models*. Transnational models are often based on learning the translations from the head entity to the tail entity (e.g., TransE) while bilinear models often tend to use a multiplicative approach and to represent the relationships as matrices in the vector space. In general, bilinear models obtain good results in the link prediction tasks [44]. Main models of this category are RESCAL [67], DistMult [103], ComplEx [88].

Translational Models We have described how TransE behaves in the previous section. Note that TransE does not efficiently learn the representations for 1-to-N relationships in a knowledge graph. This comes from how the scoring function is defined: suppose the existence of the triples (`New York City`, `locatedIn`, `State of New York`), (`New York City`, `locatedIn`, `United States`. Eventually, a scoring function consistent with $\mathbf{s}+\mathbf{p} \approx \mathbf{o}$, would make the entities `State of New York` and `United States` similar, since the elements $\mathbf{s}$ and $\mathbf{p}$ of the formula are fixed. Novel models in the translational group have been introduced to reduce the effect of this problem; we can cite in this category TransH [96] and TransR [52]. In general, translational models have the advantages of having a concise definition and getting good performances. In this same category, recent and relevant approaches are RotatE [82] and HAKE [107].

Bilinear Models RESCAL [67] is based on the factorization of the tensor (see Figure 2 and has a high expressive power due to the use of a full rank matrix for each relationship in the score function $\mathbf{h}^\intercal \mathbf{W}_r \mathbf{t}$, where the interaction between the elements comes under the form of vector-matrix products. At the same time, the full rank matrix is prone

to overfitting [107] and thus researchers that studied bilinear models have added some constraints on those representations. Indeed, DistMult [103] interprets the matrix $\mathbf{W}_r$ as a diagonal matrix, not making difference between head entity and tail entity and thus forcing the modeling of symmetric relationships [44,87]: $\phi(h,r,t) = \phi(t,r,h)$, $\forall h,t$, that force symmetry even for anti-symmetric relationships (e.g., `country`, `hypernym`).

At the same time DistMult was extended by ComplEx that models the vectors in a complex vector space to better account for anti-symmetric relationships. HolE [66] uses circular correlation, a non commutative operation between vectors, that allows us to effectively surpass the $\phi(h,r,t) = \phi(t,r,h)$ problem that DistMult had. Note that it has been proved that HolE and ComplEx are isomorphic [36]. ANALOGY [54] is a model that extends the scoring function by considering analogical relationships that exist between entities given the relationships. In their paper [54], the authors have shown that DistMult, ComplEx and HolE are special cases of ANALOGY.

Neural Models Another group with a lower number of proposed approaches consists of neural networks-based models; the Neural Tensor Network [81] is an approach for knowledge graph embeddings that uses a score function that contains a tensor multiplication, that depends on the relationship, to relate entity embeddings, this type of operation provides some interesting reasoning capabilities and was also used in later approaches as a support for reasoning using neural networks in a neural-symbolic model [80]. Instead, ConvE [18] introduces the use of convolutional layers, thus being closer to deep learning approaches. While effective, this method suffers from limited explainability and more variation given by the number of hyperparameters that increases with the number of layers [82].

Recent Approaches We hereby summarize some recent approaches that have been introduced in the literature and that are relevant with respect to the results they obtained and the ideas that stand behind them.

- Hierarchy-Aware Knowledge Graph Embedding (HAKE) [107] is one of the few models that also consider the fact that elements in the knowledge graph belong to different levels of the hierarchy (e.g., the authors use the triple *arbor/cassia/palm, hypernym, tree* as an example of elements at different levels of the hierarchy). Using polar coordinates they are able to distribute the hierarchical knowledge inside the representations.
- RotatE [82] was introduced to provide a method to effectively represent symmetric properties in knowledge graph embeddings. The authors of this paper propose to use rotation in a complex space to support symmetry and other properties. In Figure 4 we show how rotation can effectively support the definition of relationships that are symmetric; the rotation allows you to interpret symmetry as a geometric property. Authors prove that their model, implemented inside a complex vector space, can capture properties like symmetry, inversion, and composition.
- TuckER [6] is a recent approach that also uses tensor factorization for knowledge graph embeddings obtaining good results over the link prediction task.
- Another recent approach tries to apply graph convolutional neural networks to generate knowledge graph embeddings, and this might influence a new way of dealing with knowledge graph structures [79].
- Contextualized Knowledge Graph Embeddings [31] (COKE) is a method that has been inspired by recent results of contextual representation of words [68]: using

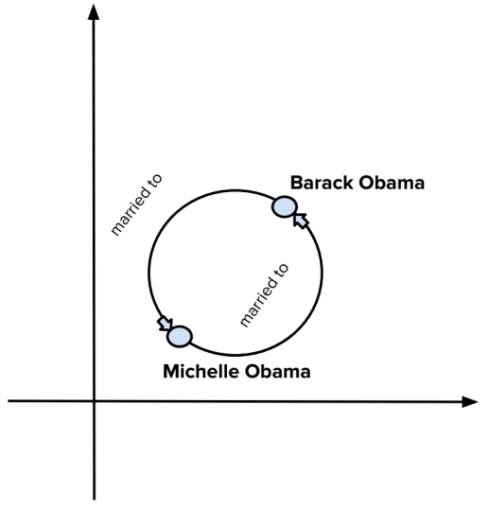

Figure 4. Example of how the use of rotation can support the definition of properties that are symmetric in the vector space. Image is adapted from [82].

transformers [89], the authors propose to capture the different meanings an entity can assume in different parts of the knowledge graph. For example, the entity Barack Obama is connected to entities related to politics, but also to the entity that represents members of his family, showing two different *contextual meanings* of the same entity. The main difference between COKE and other models is that it models the representations based on the context and thus, differently from other methods, it provides representations that are not static.

- SimplE [44] extends canonical Polyadic tensor decomposition (CP) [37] to provide good embeddings for link prediction. CP poorly performs on link prediction because it learns two independent embeddings for each entity. SimplE makes use of inverse relationships to jointly learn the two embeddings of each entity.
- Quantum embeddings [27] are a novel method to embed entities and relationships in a vector space and the representations are generated following ideas that come from quantum logic axioms [10]. These embeddings preserve the logical structure and can be used to do both reasoning and link prediction.

3.2. Enhanced Knowledge Graph Embeddings

While most of the previous approaches rely mainly on the use of the triples present in the knowledge graph to generate the vector representations; additional information (or different information) can be used inside the embeddings to generate vectors that account for a better representation. As noted by [92] attributes (like gender) need to be model in an efficient way: the attribute *male* is connected to multiple entities and thus model like TransE might not be adequate to treat this issue; in the literature, there are in fact models that have been proposed to account for better handling of these attributes [51].

Path-based Embeddings While the most common approaches use a score function that is based on triples, more recent approaches try to consider also the information that comes from a path on the graph [50,33]. There are approaches that focus on the use of Recurrent Neural Networks (RNNs) to tackle the task of multi-hop predictions [104,16].

Distributional Embeddings An alternative approach to generate embeddings comes from the computational linguistics field and it is represented by those models that view language under a distributional perspective in which the meaning of words in a language can be extracted from the usage of those words in the language. Word2vec [59] is a model that embeds words in the vector space by eventually putting words that appear in similar contexts in close positions of the vector space. In the same way, on Wikipedia using user-made links [7] or using entity linking [9] it is possible to generate embeddings of the entities of a knowledge graph using the word2vec algorithm [59]. For example, Wiki2vec[3] uses word2vec over Wikipedia text and generates the representations for both entities (by looking at links co-occurrence) and words. TEE [9] proposes to use entity linking to first disambiguate text and generate sequences of entities and then use the knowledge graph to replace the sequences of entities with sequences of most specific types; using word2vec one can generate entity and type embeddings based on the distribution in text. Methods that are based on entity linking suffer from low coverage, caused by the entity linking quality. In general, these models do not provide a direct way to embed relationships. Another prominent model in this category is RDF2Vec [72]: it uses an approach that combines techniques from the word embeddings community with knowledge graphs. It generates embeddings of entities and relationships by first creating a virtual document that contains lexicalized walks over the graph and then use word embeddings algorithm on the virtual document to create the representations.

Text-Enhanced Embeddings There instead exists a variety of models that makes use of textual information [97,23,95,100,99,41,2] to enhance the performance of knowledge graph embeddings techniques. These pre-trained representations can be used to initialize knowledge graph embeddings and to generate representations that can, in some cases, outperform other baselines [103]. As stated in the previous section, the use of textual information can be useful to generate the representations of the entities even when they are not present in the knowledge base. For example, Text-enhanced Knowledge Embedding [97] (TEKE) focuses on Wikipedia inner links and replaces them with Freebase entities and then constructs a co-occurrence network of entities and words in the text; eventually, this information is used to enrich the contextual representation of the elements of the knowledge graph. Jointly [95] is an embedding method in which textual knowledge is used to enrich the representation of entities and relationships. In this work, both entities and words are aligned into a common vector space; vectors associated with words and entities that represent a common concept are then forced to be closer in the vector space by combining different loss functions. Description-Embodied Knowledge Representation Learning (DKRL) [100] includes the description of the entities in the representation. DKRL uses a convolutional layer to encode the description of the entity into a vector representation and use this representation in the loss function. Words vectors coming from the entity description can be initialized with the use of word2vec embeddings. The model learns two representations for each entity, one that is structure-based (i.e., like TransE)

[3] https://github.com/idio/wiki2vec

and one that is based on the descriptions. One key advantage of DKRL [100] is that it offers the possibility of doing zero-shot learning of entities by using the description of the entities themselves.

Image Enhanced Embeddings Image-embodied Knowledge Representation Learning [101] (IKRL) provides a method to integrate images inside the scoring function of the knowledge graph embedding model. Essentially, IKRL uses multiple images for each entity and use the AlexNet convolutional neural network [46] to generate representations for the images; these representations are then selected and combined with the use of attention to be finally projected in the entity space, generating an image specific representation for images. Recently, approaches to exploit *multi-modal learning* on knowledge graph embeddings that combine image features and other information have also been introduced in the state-of-the-art [98,55].

Logic Enhanced Embeddings There are approaches that account for the combination of logic and facts [93,29,30,74] for knowledge representation. KALE is a model that combines facts and rules using fuzzy logic [29]. There are other approaches that try to embed knowledge graphs by keeping the logical structure consistent, we mentioned embedding with quantum axioms in Section 3.1, but there are other methods that starts with the objective of doing logical reasoning over embedded representations [80,73] (we will present more details of these approaches in the Section 4, where we discuss explainability).

Researchers have shown that it is possible to combine facts and first-order formulae using a joint optimization process. In [75], the authors propose a general approach for incorporating first-order logic formulae in embedding representations. During training, their approach samples sets of entities, and jointly minimizes the negative likelihood of the data and a loss function measuring to which extent the model violates the given rules with respect to the sampled entities. A shortcoming of this approach is that it relies on a sampling procedure, and it provides no guarantees the model will still produce predictions that are consistent with the logic rules for entities that were not observed during training. To overcome this shortcoming, in [61] authors incorporate equivalency and inversion axioms between relations by only regularizing the relation representations during the training process, where the shape of the regularizers are derived from the axiom and the model formulations. A similar idea is followed by [17] for incorporating simple implications between two relations. In [63], authors propose using adversarial training for incorporating general first-order logic rules in entity and relation representations: during training, an adversary searches for entities where the model violates the given constraints, and the model is regularized in order to correct such violations. Entities can be searched either in entity or in entity embedding space; in the latter case, the problem of finding the entity embeddings where the model maximally violates the logic rules can be efficiently solved via gradient-based optimization.

Schema-Aware Embeddings Few models in the state of the art focus on the differences between instances (i.e., entities) of a knowledge graph and concepts (like, `Country`, `City` and `Place`) [56]. Schema-rules can be useful to define constraints over score predictions. For example, they have been used to learn predicate specific parameters to decrease, in an adaptive way, the score of relationships that might be conflicting with schema rules [62].

TransC [56] proposes an interesting representation for concepts, in which each concept is represented as a sphere and each entity is a vector. An instance-of relationship can be easily verified by checking if the entity is contained inside the sphere. In one of the previous sections, we mentioned HAKE (Hierarchy-Aware Knowledge Embeddings) [107] as a recent method that considers the hierarchical topology in the embedding. This aspect is also important in the context of analysis over explainability: modeling ontologies is a needed step to learn how to model logical reasoning and provide justifiable inferences, however, not all methods are capable of modeling rules [32].

There are also approaches that considers the fact that the ontology can be used to provide better representations, for example Type-embodied Knowledge Representation Learning (TKRL) [102]. Given a triple h,r,t, the subject $\mathbf{h}$ and the object $\mathbf{t}$ are projected to the type spaces of this relation as $\mathbf{h}_r$ and $\mathbf{t}_r$, the projection matrices become type-specific. TKRL optimizes the following scoring function: $||\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r||$. In this group we also include TRESCAL [14] an extension of RESCAL [67] that considers types in the tensor decomposition. On the other hand, there do exist approaches that generate the representations of ontology concept by taking in consideration the co-occurrence of types in text [9].

Hyperbolic Embeddings Many approaches in the state-of-the-art rely on the use of representations in the Euclidean space. However, when dealing with the representations of tree-like structures (e.g., some ontologies can be interpreted as trees) Euclidean spaces have to rely on many dimensions and are not suited to represent trees. Euclidean geometries rely on Euclid's axiom of the parallel lines, but there do exist other geometries that do not consider it. Hyperbolic geometries allow us to use hyperbolic planes where trees can be effectively encoded. These approaches have been now widely used to represent tree-like structure [65,83,77] and received recognition in natural language processing [47,85,90]. In general, these approaches have been applied to ontological trees (e.g., the WordNet hierarchy) and cannot account for knowledge graph structures that are more complex. Recently, embedding in the hyperbolic plane has shown to be effective also for knowledge graphs [5,45] since they can provide better ways to model topological structures [45].

Temporal Knowledge Graph Embeddings There are also approaches that are meant to account for temporality in knowledge graph embeddings by considering temporal link prediction (i.e., consider that some predicates, like *president of*, have values that change over time) and to study the evolution of knowledge graphs over time [42,22,25]. For example, recurrent neural networks can be used to learn time-aware relation representations [25].

3.3. Evaluation and Replication

Evaluation in knowledge graph embeddings is often based on link prediction. In general, the link prediction task can be defined as the task of finding an entity that can be used to complete the triple $(h,r,?)$; for example, (`New York City`, `country`, `?`), where `?` is United States. To compute the answer for the incomplete triple generally the score function is used to estimate the *likelihood* of the entities. The procedure is the following: for each triple to test, we remove the head and we compute the value of the score function for each of the entities that we have in the dataset and we rank them from higher to

Dataset	# Entities	# Relations	Train	Validation	Test
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,505	237	272,115	17,535	20,466
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	2,824	2,924
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Table 2. Number of entities, relationships and training, validation, test triples for the main dataset used in the state-of-the-art.

lowest. Then we collect the rank of the correct entity. The same is done by replacing the tail of the triple. At the end, the average rank is computed, this measure is called Mean Reciprocal Rank (MRR). Another measure that is often used in the link prediction setting is the HITS@K (with K commonly in $1, 3, 10$).

[11] uses a *filtering* setting that has become a standard of the evaluation. The evaluation of the MRR is influenced by the fact that some *correct* triples share entity and relationship (e.g., (`United States`, `countryOf`, `?`) is true for multiple triples) and they can be ranked one over the other in the ranking list, thus biasing the results. What it is typically done when computing the MRR for a triple in this setting is to filter out the other triples that are true and that are present in the training/validation/test set.

FB15k [11] is a subset of Freebase while WN18 [11] is a Word-Net subset. FB15k and WN18 were both introduced in [11] and originally come with a training, validation and test split.

The quality of these two datasets has been argued in more recent work [86,18]. FB15k originally contained triples in the test set that are the inverse of those present in the training set, for example `/award/award_nominee` and `/award_nominee/award`. While those links are not false, they could bias the results by making the task easier for learning models (i.e., models can just learn that one relationship is the inverse of the other [86], and models that force symmetry, like DistMult, could perform better just because of the dataset used). The same problem was found in WN18 [18]. This brought researchers to introduce two novel datasets, a subset of the original ones, that do not contain easy-to-solve cases. FB15k-237 has been introduced by [86] and WN18RR was introduced by [18] and they are a subset of FB15K and WN18 respectively. Take into account that the DistMult model favored the symmetry between the relationships.

YAGO3-10 [57,18] has recently become quite popular, it contains a subset of the YAGO knowledge graph that consists of entities that have more than 10 relationships each. As noted by [18] the triples in this dataset account for descriptive attributes of people (e.g., as citizenship, gender, and profession). Another really important dataset is Countries [12], which is often used to evaluate how well knowledge graph embeddings learn long term logical dependencies. Note that while in general, the datasets used are the ones we described, some papers introduce new datasets when needed. For example, a subset of the YAGO dataset (namely YAGO39K) has been used to evaluate TransC a work that extended embeddings with the use of concepts [56].

In Table 2 we show numerical data related to these datasets. It is important to notice that these datasets are small with respect to the size of knowledge graphs (e.g., DBpedia has more than 4 million entities).

Link prediction is not the only task on which knowledge graph embedding are evaluated, often the evaluation takes into account the task of triple classification, that is the