

CaLiGraph: A Knowledge Graph from Wikipedia Categories and Lists

Nicolas Heist^a and Heiko Paulheim^{b,*}

^a *metaphacts GmbH, Germany*

E-mail: nh@metaphacts.com

^b *University of Mannheim, Germany, Data and Web Science Group*

E-mail: heiko.paulheim@uni-mannheim.de

Editors: First Editor, University or Company name, Country; Second Editor, First University or Company name, Country and Second University or Company name, Country

Solicited reviews: First Solicited reviewer, University or Company name, Country; anonymous reviewer

Open review: First Open Reviewer, University or Company name, Country

Abstract. Knowledge Graphs (KGs) are increasingly used for solving or supporting tasks such as question answering or recommendation. To achieve a useful performance on such tasks, it is important that the knowledge modeled by KGs is as correct and complete as possible. While this is an elusive goal for many domains, techniques for automated KG construction (AKGC) serve as a means to approach it. Yet, AKGC has many open challenges, like learning expressive ontologies or incorporating long-tail entities. With CaLiGraph, we present a KG automatically constructed from categories and lists in Wikipedia, offering a rich taxonomy with semantic class descriptions and a broad coverage of entities. We describe its extraction framework and provide details about its purpose, resources, usage and quality. Further, we evaluate the performance of CaLiGraph on downstream tasks and compare it to other popular KGs.

Keywords: CaLiGraph, Ontology Construction, Knowledge Graph Population, Knowledge Graph Evaluation, DBpedia, YAGO

1. Introduction

1.1. Motivation and Problem Statement

An essential property of a truly intelligent application is its ability to access all the information necessary to solve the task it is designed for. With the advent of Knowledge Graphs (KGs), this long-standing objective in AI of supplying machines with relevant information is gradually becoming a reality [1, 2]. KGs are the key technology to tie together data and knowledge [3]. Thereby, they diminish the effort of combining data with other sources [4] or using it in applications of various domains (e.g., agriculture [5], manufacturing [6], or tourism [7]) and task types (e.g., advertising [8], question answering [9] or recommendation [10]).

The core idea of KGs is to represent data as a labeled directed graph, with nodes representing concepts or concrete instances and edges representing relations between them.¹ Using graphs to represent data has several advantages

*Corresponding author. E-mail: heiko.paulheim@uni-mannheim.de.

¹Weikum [2] points out that *Knowledge Base* would be the proper term as a graph struggles to express higher-arity relations or constraints natively. In this work, we use the term *Knowledge Graph* inclusively as it is the more common terminology.

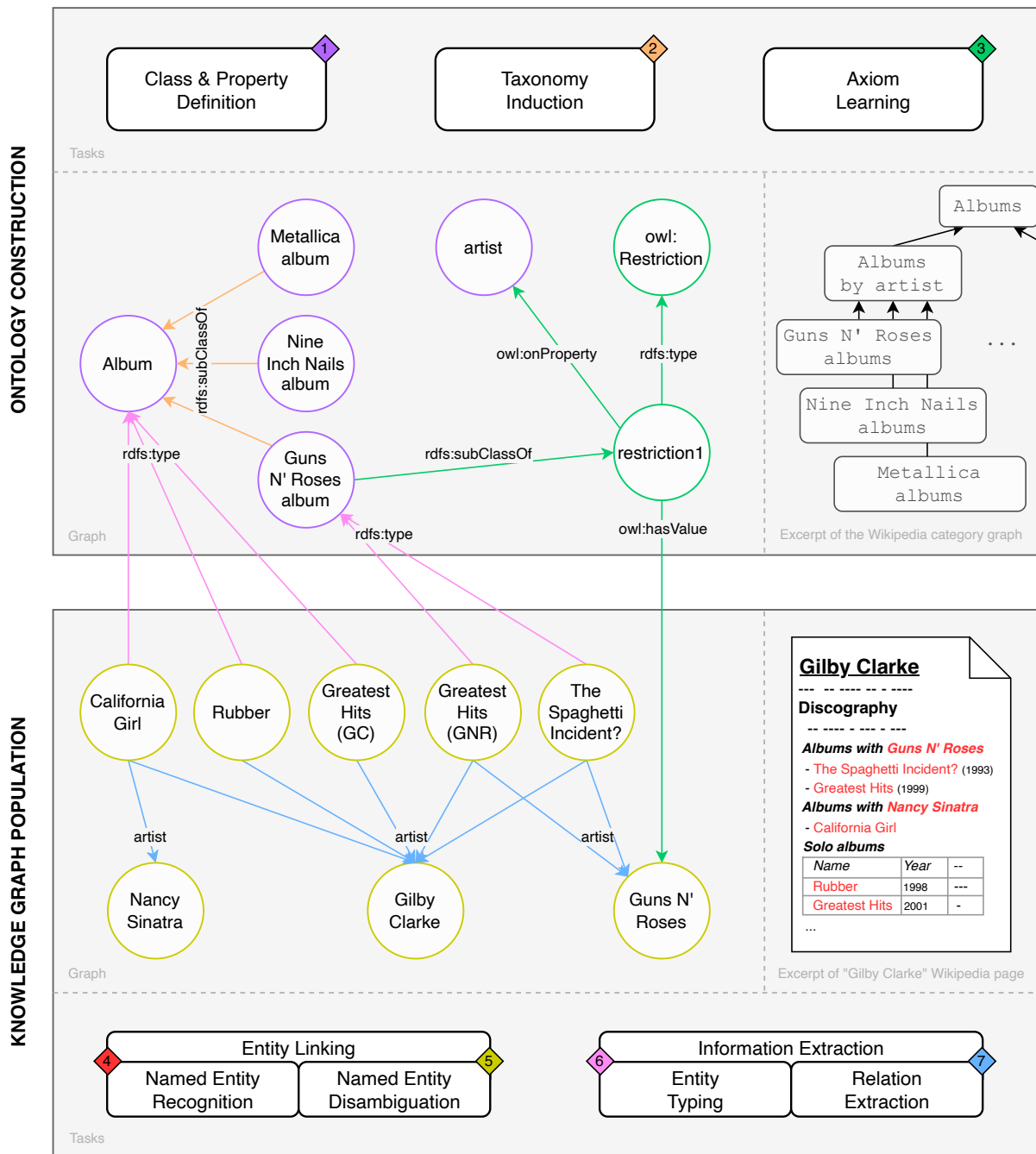


Fig. 1. An overview of the typical steps addressed during Knowledge Graph construction.

over relational or NoSQL alternatives, like the flexible definition and reuse of schemas and the large variety of graph-based techniques for querying, search or analytics [11]. As shown in Figure 1, nodes in a KG may represent concepts (e.g., the class *Album* or the relation *artist*) or entities (e.g., the album *California Girl* or the artist *Nancy Sinatra*). Relations may exist between concepts (*Guns N' Roses album* is a sub-class of *Album*), between a concept and an entity (*California Girl* is an *Album*) or between entities (*California Girl* has the artist *Nancy Sinatra*). All this information is typically stored in the form of (*subject, predicate, object*) triples.

The trend of entities added to publicly available KGs in recent years indicates they are far from complete. The number of entities in Wikidata [12], for example, grew by 26% in the time from October 2020 (85M) to October 2023 (107M).² Wikidata describes the largest number of entities and comprises – in terms of entities – other public KGs to a large extent. [13] Consequently, this challenge of incompleteness applies to all public KGs, particularly when it comes to long-tail and emerging entities [14].

Automatic information extraction approaches can help mitigate this problem if the approaches ensure that the extracted information is high quality. While the performance of open information extraction systems (i.e., systems that extract information from general web text) has improved in recent years [15–17], the quality of extracted information has not yet reached a level where integration into public KGs like DBpedia [18] should be done without further filtering.

The extraction of information from semi-structured data is, in general, less error-prone and already proved to yield high-quality results as, for example, DBpedia itself is extracted primarily from Wikipedia infoboxes; further approaches use the category system of Wikipedia [19, 20] or focus on tables (in Wikipedia or the web) as semi-structured data source to extract entities and relations [21]. As highlighted by Weikum [2], first "picking low-hanging fruit" by focusing on premium sources like Wikipedia to build a high-quality KG is crucial as it can serve as a solid foundation for approaches that target more challenging data sources.

1.2. Contributions

We present CaLiGraph, a KG automatically constructed from semi-structured content in Wikipedia. CaLiGraph uses DBpedia as a foundation to extract an extensive taxonomy from the category graph in Wikipedia and enriches it with OWL-based axioms describing the semantics of the classes. Further, it uses various information extraction techniques to extract new entities and facts from enumerations and tables in Wikipedia, particularly focusing on constructs where similar entities co-occur. In its most recent version, CaLiGraph describes 1.3 million classes and 13.7 million entities.

In this work, we give a comprehensive overview of CaLiGraph. In particular, our contributions are as follows:

- We give an overview of the field of automated KG construction and formulate open challenges in Section 2.
- We summarize the extraction process of CaLiGraph, including all relevant inputs, in Section 3.
- We describe the purpose, contents, resources and use cases of CaLiGraph in Section 4.
- We provide statistics, quality metrics and evaluations of the major CaLiGraph versions as well as comparisons to popular public KGs in Section 5.

2. Automated Knowledge Graph Construction

The most straightforward way to create a KG is through manual definition. Cyc [22] and WordNet [23] are notable examples, employing a team of experts to insert the data by hand. While this is feasible for domains with a manageable amount of data, the potential to scale up is very limited. [24] Freebase [25] and, more recently, Wikidata [12] are examples of achieving scalability in manual curation via crowd-sourcing.

In this paper, we only consider automatically extracted KGs in this work. Apart from manually curated KGs, this excludes KGs relying on human-in-the-loop mechanisms [26] or dataset-dependent RML mappings [27, 28] to extract instance data.

Generally, automated KG construction (AKGC) can use various types of input data. The vast majority of approaches work on unstructured texts, i.e., tries to extract triples from plain text. For example, from the sentence *Trent Reznor produced Marilyn Manson's album "Antichrist Superstar"*, we can derive triples like

< Antichrist_Superstar, producer, Trent_Reznor >

< Antichrist_Superstar, artist, Marilyn_Manson >

²<https://tools.wmflabs.org/wikidata-todo/stats.php>

The family of approaches that directly extract such triples without a predefined schema or set of entities is called *Open Information Extraction*, with notable examples being TextRunner, ReVerb, and OLLIE [29]. Other approaches, such as NELL, predefine a set of relations to be extracted, but do not limit the set of entities [30], while a third family limits the set of entities to a given knowledge graph, which makes it fall into the area of knowledge graph refinement [31]. Most of these approaches use a combination of different building blocks, such as coreference resolution, triple extraction, entity linking, and relation linking [32].

In the following, we present an AKGC implemented in the CaLiGraph extraction framework. We use the pipeline to compare popular KGs on the web and formulate challenges and limitations in AKGC. CaLiGraph has multiple differences to the aforementioned approaches:

- CaLiGraph relies on a corpus of semi-structured documents (i.e., listings and categories in Wikipedia). In all those lists, entities are typically mentioned only once, using an informative label. This makes the linguistic part of the processing easier, as it does not require semantic parsing or coreference resolution.
- CaLiGraph uses a given knowledge graph as its backbone, but extends both its ontology as well as the set of entities. This makes it different to many other AKGC pipelines, which keep the ontology and/or the set of entities fixed.

2.1. A Pipeline for Automated Knowledge Graph Construction

KG construction is typically not an end-to-end ML task but consists of multiple steps, each with unique requirements and challenges [2]. Figure 1 lists the steps in the order they are addressed in the CaLiGraph extraction framework, together with actual examples. The pipeline consists of the two high-level blocks of *Ontology Construction* (OC) and *Knowledge Graph Population* (KGP), with the former being responsible for the definition of the ontology necessary to describe the domain (the so-called T-box) and the latter being responsible for populating the graph with data using concepts of the ontology (the A-box). Whether the steps are executed once or iteratively, in this sequence or another, depends on the KG to be extracted.

Ontology Construction steps:

1. **Class & Property Definition** Define relevant classes and properties of the domain
2. **Taxonomy Induction** Discover hierarchical relationships among classes and properties
3. **Axiom Learning** Formulate constraints for classes (e.g., disjointnesses) and properties (e.g., domains/ranges)

Knowledge Graph Population steps:

4. **Named Entity Recognition** Identify mentions of named entities in a given data corpus
5. **Named Entity Disambiguation** Add the mentions to the KG by creating new or updating existing entities
6. **Entity Typing** Discover type assertions for the entities in the KG using the data corpus
7. **Relation Extraction** Discover relation assertions for the entities in the KG using the data corpus

While the steps in the *Ontology Construction* block may be conducted manually for a sufficiently small domain, the steps in the *Knowledge Graph Population* block are always automated processes using a given data corpus.

2.2. Knowledge Graphs on the Web

Given the pipeline above, we discuss the construction processes of automatically extracted general-purpose KGs. We only consider publicly accessible KGs and disregard closed-source industry-created KGs like those from Microsoft, Facebook, Amazon or ebay [33]. Figure 2 shows a timeline with the major milestones of the public KGs discussed in the following.

2.2.1. DBpedia

DBpedia [18] aims to represent the knowledge of Wikipedia in a structured form and focuses on infoboxes to extract knowledge.

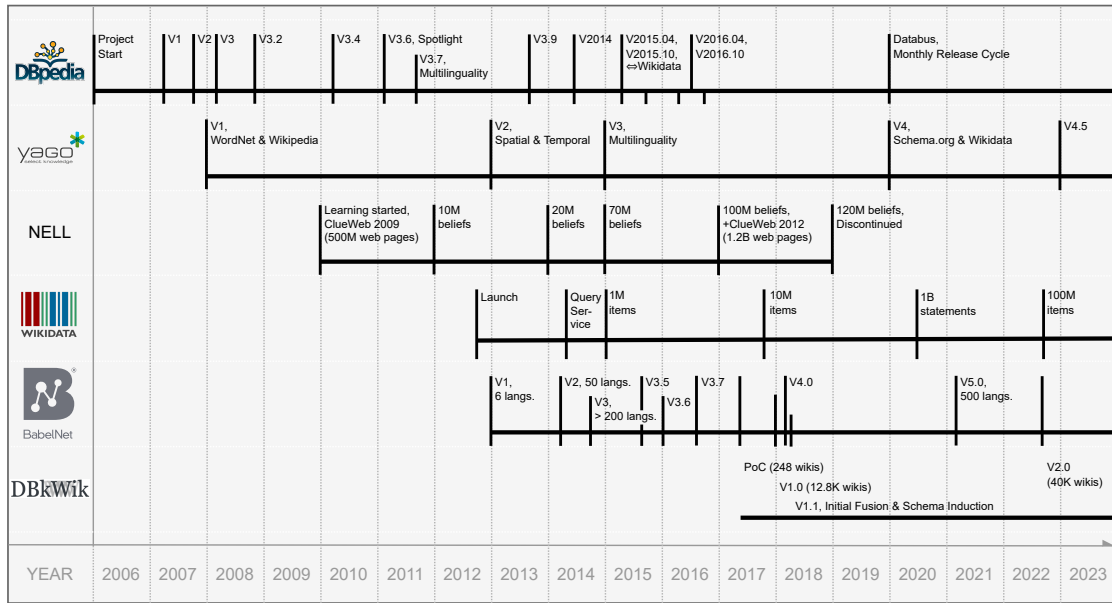


Fig. 2. A timeline with major milestones of popular public KGs.

Ontology Construction DBpedia provides a Mappings Wiki³ where the community defines classes, properties, datatypes and restrictions. Further, they map infoboxes to types in the schema and infobox keys to properties.

Knowledge Graph Population DBpedia defines one entity per article in Wikipedia. A disambiguation of entities is unnecessary as they are marked with hyperlinks in the text. Type assertions are derived from infobox types, and relation assertions are derived from infobox keys.

2.2.2. YAGO

YAGO [34] is built on the idea of combining a small but well-crafted top-level schema with a large but messy taxonomy, thereby creating a unified and cleaned schema. Further, they tap other data sources to ingest additional data from various domains.

Ontology Construction Up to version 3 [19], YAGO automatically combines WordNet [23] with the Wikipedia category graph to create a large ontology. They add axioms for some classes derived from the category graph using hand-crafted rules. In version 4 [35], they fundamentally change the KG by combining the ontology from Schema.org [36] with the one from Wikidata to create a cleaned, “reason-able” version of Wikidata. They define manual mappings between Schema.org classes and Wikidata classes to create the combined ontology and add rudimentary SHACL constraints to ensure data validity.

Knowledge Graph Population Up to version 3, YAGO performs KGP similarly to DBpedia, using articles as entities and extracting assertions from infoboxes. Additionally, they define an enhancement process where additional entities may be added from any external / sources or tools. In version 2 [37], temporal and geospatial data is integrated, and in version 3 [19], multilingual data from multiple Wikipedia language chapters is added. In version 4, entities and assertions are taken from Wikidata.

2.2.3. NELL

NELL [38] is an example of extracting a KG from free text. It was originally trained with a few seed examples and continuously ran an iterative coupled learning process. In each iteration, facts were used to learn textual patterns to

³<https://mappings.dbpedia.org>

1 detect those facts, and patterns learned in previous iterations were used to extract new facts, which served as training 1
 2 examples in later iterations. NELL introduced a feedback loop incorporating occasional human feedback to improve 2
 3 the quality. 3

4 *Ontology Construction* NELL started with an initial ontology defining hundreds of concepts and binary relations. 4
 5 During runtime, the ontology is extended with additional concepts and relations. 5
 6 6

7 *Knowledge Graph Population* NELL is bootstrapped with a dozen examples for each concept and relation. New 7
 8 entities and assertions are added with each iteration. 8
 9 9

10 2.2.4. BabelNet 10

11 BabelNet [39] is a KG that integrates encyclopedic and lexicographic knowledge from Wikipedia and WordNet 11
 12 in multiple languages. 12

13 *Ontology Construction* The ontology consists of concepts derived from senses in WordNet and from articles and 13
 14 categories in Wikipedia [40]. They connect the two resources by mapping senses to articles automatically. In early 14
 15 versions, only lexical properties are used. In the recent version, they integrate related KGs like Wikidata and YAGO, 15
 16 taking over their semantic properties as well. 16

17 *Knowledge Graph Population* Initially, the graph was populated with entities from Wikipedia articles. From Word- 17
 18 Net, lexical and semantic pointers between synsets are extracted as relations. Relations between Wikipedia articles 18
 19 were initially extracted as unlabeled relations. In the recent version, there are efforts to extract the semantics of the 19
 20 relations. Further, assertions from related KGs like Wikidata and YAGO are included [41]. 20
 21 21

22 2.2.5. DBkWik 22

23 DBkWik [42] aims to extract and fuse data from thousands of Wikis of arbitrary content from a Wikifarm, for 23
 24 example, Jedipedia⁴ or Music Hub.⁵ 24

25 *Ontology Construction* DBkWik uses a variation of the DBpedia extraction framework to extract data from Wikis. 25
 26 Contrary to DBpedia, DBkWik has no community-defined mappings. Instead, they generate a shallow schema from 26
 27 the infoboxes of each Wiki and fuse these schemas afterwards. Then, they enrich the unified schema with subclass 27
 28 relations and restrictions for domains and ranges. 28
 29 29

30 *Knowledge Graph Population* Entities are derived from articles in the Wikis, and assertions are derived from 30
 31 infoboxes. Similar to the schema, entities must also be matched to avoid duplicates from overlapping Wikis. 31
 32 32

33 2.2.6. Wikidata 33

34 Wikidata [12] is, to date, one of the largest publicly available knowledge graphs. The original motivation was to 34
 35 unify the information in infoboxes across different language editions of Wikipedia by storing the main information 35
 36 about entities in a central knowledge graph. 36

37 *Ontology Construction* Wikidata uses the crowd sourcing approach for classes and properties as well. The ontol- 37
 38 ogy is maintained by the Wikidata user base. 38

39 *Knowledge Graph Population* Entities can be entered and altered by users of Wikidata. Moreover, larger data 39
 40 dumps (called *data donations* by Wikidata) are imported, such as national library data.⁶ In addition to users, a 40
 41 number of specialized bots create and update entities, usually by incorporating updates on public databases, such as 41
 42 publication databases, in Wikidata. 42
 43 43

44 2.3. Limitations and Challenges 44

45 45
 46 In Table 1, we list the advantages and limitations of the previously discussed KGs. Following, we distill these 46
 47 into a (incomplete) list of challenges (mostly complementary to challenges mentioned by Weikum [2]): 47
 48 48

49 ⁴<https://jedipedia.fandom.com/> 49

50 ⁵<https://music.fandom.com/> 50

51 ⁶https://www.wikidata.org/wiki/Wikidata:Data_donation 51

Table 1
Advantages and Limitations of public general-purpose KGs.

Name	Advantages	Limitations
DBpedia	The ontology is hand-curated and, hence, of high quality. Entities and assertions are extracted from highly structured data and are of high quality as well. Due to its pioneering role of representing Wikipedia and its good accessibility, DBpedia serves as a central hub of the linked data web. ⁷	The manually-defined schema has limited expressiveness and flexibility. DBpedia is biased towards popular entities as Wikipedia allows articles only if the subject is of a certain notability. ⁸ Further, the information in the KG is limited to the content of the infoboxes in Wikipedia.
YAGO	Both YAGO3 and YAGO4 have very expressive and fine-grained ontologies. Due to many external sources and the connection to Wikidata, YAGO has a high density of assertions per entity.	YAGO3 suffers from the same problem of representing tail entities as DBpedia. In YAGO4, many more entities are ingested through the switch to Wikidata, which again introduces the limitation of manual curation. What happens to the manually defined mappings if Schema.org and/or the Wikidata taxonomy change is unclear.
NELL	The coverage of schema, entities and assertions in NELL is limited only by the available information in the data source consisting of a Web crawl.	The quality is comparably low as NELL initially starts with very little knowledge and uses only web resources with occasional human feedback during knowledge acquisition. In a link prediction evaluation, NELL scored a MAP of 0.35 in 2010 and 0.55 in 2017 [38]. Refinement iterations are run on a fixed Web crawl, i.e., recent knowledge is not considered.
BabelNet	BabelNet emphasizes the lexicographic perspective, not only describing entities but also the senses of the words that entities are referenced with.	While a large variety of resources is included by exploiting mappings to other KGs, long-tail entities and new properties are not explicitly addressed.
DBkWik	DBkWik manages to tap additional data sources by targeting thousands of Wikis from a Wikifarm and can hence integrate specialized knowledge from many different domains.	Creating a comprehensive schema from thousands of Wikis is a difficult task, especially when little information about the individual concepts and entities is available. DBkWik is restricted to entities defined in the ingested Wikis.
Wikidata	Wikidata does not only contain data, but also provenance information and other metadata, thereby making it a very trustworthy source.	Since it can be directly edited by everybody, Wikidata is more susceptible to vandalism than other knowledge graphs. [43] Its non-standard data model poses challenges to some downstream applications.

- Coverage of unknown properties.** Many KGs use a fixed schema with pre-defined properties to model knowledge. Extending this schema is a challenging task; potential errors greatly impact the KG quality.
- Coverage of long-tail entities.** Identifying and disambiguating long-tail entities is difficult as the data source contains, by definition, only limited information about them; most KGs choose to use a fixed set of entities like the set of Wikipedia articles.
- Ontology with expressive, fine-grained types.** An expressive and detailed ontology is a prerequisite for comprehensive data modelling. While some KGs have very detailed taxonomies already, there is a lack of axioms that explicitly describe the intent of their types.
- Maintaining high data quality.** The quality of automatically constructed KGs will never be perfect, but KGs should have a certain quality to be useful in downstream tasks (typically, a correctness of 95% is desired [2]). Apart from the methods used, quality depends on the data sources targeted during extraction.

We created CaLiGraph to tackle several of these challenges in the context of Wikipedia. We exploit semi-structured data structures like listings and tables to extract information about novel entities (C2). We create a schema from the Wikipedia category graph and enrich it with semantic restrictions describing the meaning of the concepts (C3). All the automated extraction procedures target structured or semi-structured data to minimize errors and ensure high extraction quality (C4).

⁷<https://lod-cloud.net/>

⁸<https://en.wikipedia.org/wiki/Wikipedia:Notability>

3. The CaLiGraph Extraction Framework

This section describes the extraction framework of CaLiGraph with respect to the tasks shown in Figure 1. Section 3.1 provides details about the parts of Wikipedia used in the extraction process, Section 3.2 describes how the ontology is created and Section 3.3 describes the KG population process. This section intends to give a crisp overview of the complete construction process of CaLiGraph without going into detail too much. We provide references to additional material within the section for the interested reader.

3.1. Wikipedia as Semi-Structured Data Source

Due to its structured and encyclopedic nature, Wikipedia provides interesting conditions to extract information automatically. Concretely, we select Wikipedia as a data corpus for CaLiGraph as it has several advantages:

Structure Wikipedia is written entity-centric with a focus on facts. Due to the encyclopedic style and the crowd reviewing process, it has a fairly consistent structure. Wikipedia uses its own markup language, i.e., wikitext or Wiki markup⁹, which allows for more concise access to (semi-)structured page elements such as sections, listings, and tables, compared to plain HTML. Listings are often used to provide an overview of a set of entities that are related to the entity an article is about. Section titles are typically used consistently for specific topics (e.g., for the Discography of a band).

Entity Links If a Wikipedia article is mentioned in another article, it is typically linked in the Wiki markup (a so-called *blue link*). Furthermore, it is possible to link to an article that does not (yet) exist (a so-called *red link*). As Wikipedia articles can be trivially mapped to entities in Wikipedia-based KGs like DBpedia, since they create one entity per article, we can identify many named entities in listings and their context without the help of an entity linker.

Access Wikipedia snapshots are published periodically as XML dumps that can be processed conveniently. Many high-quality open-source libraries exist for the interpretation of Wiki markup. In our framework, we use WikiTextParser¹⁰ to process the markup in Python.

DBpedia With DBpedia [18], a well-established Wikipedia-based KG is already available. As it is extracted primarily from infoboxes, the information in DBpedia is very accurate and thus a perfect source for distant supervision [44].

In the remainder of this section, we briefly explain the main Wikipedia elements exploited to extract CaLiGraph. Provided statistics are computed on the Wikipedia dump the most recent CaLiGraph version 3.1.1 is based on (August 2022, English).

3.1.1. Articles

An article in Wikipedia describes a concept of the real world. In the following, we will refer to this concept as the *main entity* of the article. Articles typically start with an abstract that summarizes the main entity's key facts, followed by sections about notable details. For example, the article about *Gilby Clarke* contains two sections about his career and discography.

Wikipedia contains 6.1 million articles in English (excluding non-encyclopedic pages like disambiguation pages and redirects).

3.1.2. Listings

With listings, we refer to (semi-)structured elements in Wikipedia that contain several items. In many cases, a listing represents a concept, with each item describing a concrete instance of this concept.¹¹ We are particularly interested in listings with items explicitly mentioning the entities they describe. We refer to these entities as *subject entities (SEs)*, and we define them as *all entities in a listing appearing as instances to a common concept* [46].

⁹<https://en.wikipedia.org/wiki/Help:Wikitext>

¹⁰<https://github.com/5j9/wikitextparser>

¹¹For counter-examples, we point to our previous work [45].

Name	Township(s)	Coordinates	NTS map	Status	CGNDB id
Jackpine Lake	Banting, Chambers	47°8'44"N 79°56'3"W	031M/04	Official	FBRBM
James Lake	Best	47°10'41"N 79°44'26"W	031M/04	Official	FBRHD
Jamieson Lake	Banting	47°9'22"N 79°59'37"W	031M/04	Official	FBRHY
Jessie Lake	Strathcona	47°2'28"N 79°48'14"W	031M/04	Official	FBRSJ
Jumping Cariboo Lake	Law, Olive	46°52'57"N 79°46'32"W	031L/13	Official	FBSOZ
Jumpingcat Lake	Belfast, Joan	47°1'48"N 80°9'59"W	041P/01	Official	FBSPA

(a) A lake in Ontario, CA.
Lakes of Temagami

- [Elbow Lake](#), [46°21'53"N 113°01'31"W](#), el. 7,746 feet (2,361 m)^[14]
- [Evans Lake](#), [47°00'15"N 113°04'19"W](#), el. 4,193 feet (1,278 m)^[15]
- [Hagan Pond](#), [46°28'46"N 112°52'55"W](#), el. 5,000 feet (1,500 m)^[16]
- **James Lake**, [47°04'32"N 113°12'43"W](#), el. 4,137 feet (1,261 m)^[17]
- [Jones Lake](#), [47°02'35"N 113°08'35"W](#), el. 4,088 feet (1,246 m)^[18]
- [Kleinschmidt Lake](#), [46°58'33"N 113°02'35"W](#), el. 4,186 feet (1,276 m)^[19]

(b) A lake in Montana, US.
List of lakes of Powell County, Montana

Members [edit]

- Lionel Williams – vocals, various instruments (2007–present)

Associated musicians [edit]

- Bryan Lee – drums (2007–2010)
- Calin Stephensen – bass (2007–2008)
- **James Lake** – drums/synth (2011–2017)
- Ian Gibbs – various instruments (2011–2018)

(c) A musician in the band Vinyl Williams.
Vinyl Williams

Character	Actor/Actress	Duration
Joe Lacerra	Stephen Liska	1998–2005
Cindy Lake	DeAnna Robbins	1982–83
James Lake	Glenn Corbett	1983
Mary Margaret Lake	Fawne Harriman	1983
Sammy Lake	Danny McCoy Jr.	1978
Hilary Lancaster	Kelly Garrison	1991–93
Dr. Joshua Landers	Heath Kizzier	1996–98

(d) A character in a soap opera.
List of The Young and Restless characters

Fig. 3. Listings in Wikipedia containing the mention James Lake [47]. All of the mentions refer to distinct entities; a dedicated Wikipedia article exists only for the entity of the mention in (a).

In Figure 3, we show four different listings in the form of tables (Figures 3a and 3d) and enumerations (Figures 3b and 3c). For example, in Figure 3d, the soap opera characters are considered SEs, while the actors are not, as the listing focuses on the characters. While listings are usually formatted as enumerations or tables, they have no convention of how their information is structured. For example, SEs can be listed somewhere in the middle of a table (instead of in the first column), and enumerations can have multiple levels. Further, SEs may already be marked as entities through Wiki markup (blue or red links), but this is not always true.

Of the 6.1 million articles in Wikipedia, 2.1 million contain at least one listing in the form of an enumeration or a table. We find 3.5 million enumerations and 1.4 million tables in these articles.¹² On average, listings have 11.7 items with a median of 7.

3.1.3. List Pages

List pages are a special kind of Wikipedia pages that serve the sole purpose of listing entities with a common property. The list page *List of lakes of Powell County, Montana* contains an enumeration of all lakes in *Powell County, Montana*, together with their coordinates and elevation (see Figure 3b for a snippet of the page). In contrast to arbitrary listings in Wikipedia, listings in list pages are relatively easy to exploit as they explicitly serve the purpose of listing SEs.

Wikipedia contains 89K list pages with 159K tables and 381K enumerations. On average, listings have 21.8 items with a median of 9 items.

3.1.4. Categories

Contrary to list pages, categories are a formal construct in Wikipedia and serve the purpose of categorizing pages in a hierarchical structure. This structure, the Wikipedia Category Graph (WCG), is a directed but not acyclic graph. It does not only contain categories used for categorising articles but also ones used for administrative purposes (e.g., the category *Wikipedia articles in need of updating*). The WCG has been used extensively for taxonomy induction (e.g. in [19, 40]) and has yielded highly accurate results. A subgraph of the WCG contains list categories,¹³ which organizes many of the list pages in Wikipedia. The list page *List of lakes of Powell*

¹²These numbers exclude very small listings with less than three items, which we do not consider.

¹³A list category is a Wikipedia category that starts with the prefix *Lists of*.

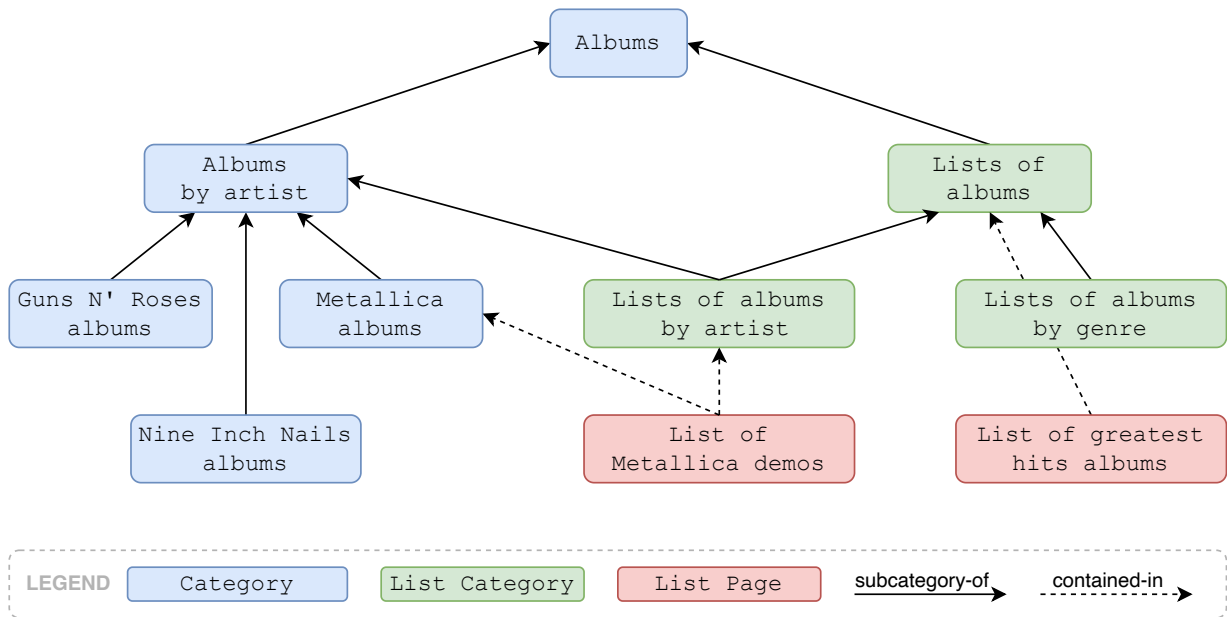


Fig. 4. Hierarchical relationships between categories, list categories and list pages in Wikipedia.

County, Montana, for example, is a member of the list category Lists of lakes of Montana by county, which in turn has the parent list category Lists of lakes of the United States, but also the parent category Lakes of Montana by county. List pages, list categories, and categories in general are tightly interconnected.

Wikipedia contains 2.2 million categories, with 11K list categories and 311K categories used for non-encyclopedic purposes like maintenance. We regard categories as of the latter kind if they are no transitive subcategories of the category Main topic classifications or have one of the following keywords in their name: *wikipedia*, *lists*, *template*, *stub* [48].

3.2. Ontology Construction

In the following, we explain how the CaLiGraph extraction framework builds an ontology from categories and lists in Wikipedia [49] and how the classes are enriched with expressive axioms [48] (cf. upper part of Figure 1).

3.2.1. Class & Property Definition

All encyclopedic categories and list pages in Wikipedia are considered candidate classes for the CaLiGraph taxonomy. Additionally, we reuse and link to classes of the DBpedia ontology. By doing so, we can effortlessly enrich CaLiGraph with additional parts of the DBpedia ontology, like relations and disjointness axioms.

The category candidates contain many categories that are suitable classes for a taxonomy like Albums or People from London. However, many non-taxonomic categories are contained as well. For example, the category London comprises all London-related topics like the pages London and Outline of London. To remove non-taxonomic categories, we rely on the observation made by Ponzetto and Navigli [50] that a Wikipedia category is a valid type in a taxonomy if its head noun is in the plural. Consequently, we identify the head nouns of the categories and remove all categories with singular head nouns.¹⁴

3.2.2. Taxonomy Induction

After removing non-taxonomic categories, we first build a taxonomy from the remaining categories, list categories, and list pages. To combine those, we use the existing connections in Wikipedia. Figure 4 shows an example

¹⁴We use spaCy (<http://spacy.io>) for head noun tagging.

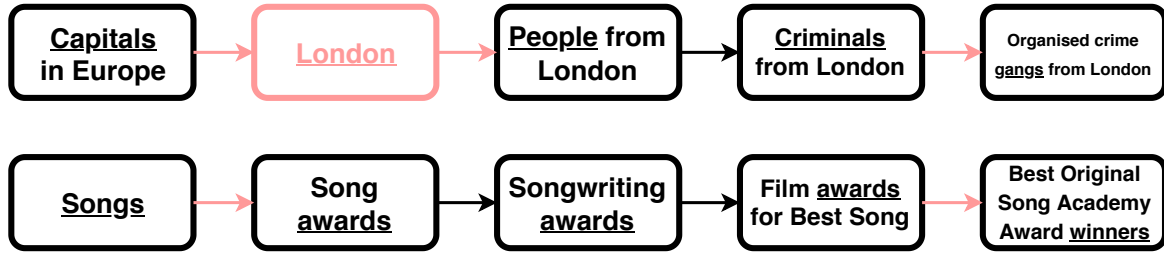


Fig. 5. Examples of non-taxonomic nodes and edges (marked in red) that must be removed from the respective category graph or list graph. [49]

of how these groups are connected. While all edges in the figure could be used to form the taxonomy, some edges should be discarded. For example, the category *Albums* has the subcategory *Album covers*, and the category *Lists of albums* contains the list page *List of controversial album art*. We remove such edges by considering the head nouns of the parent and the child: if the head noun of the parent is a synonym or a hypernym of the child's head noun, we keep the edge; otherwise, we discard the edge (as is the case for *cover* vs. *album* and *art* vs. *album*). We retrieve synonyms and hypernyms from multiple sources, for example, by crawling Wikipedia article texts for Hearst patterns [51] and from WebIsALOD [52].

Removing wrong class subsumption axioms is a crucial step for different reasons. First, the Wikipedia category graph is not acyclic, but a class hierarchy should be. In fact, some of the subcategory axioms are questionable, such as making *Capitals in Europe* a subcategory of *London* (and not vice versa), as shown in the upper part of Figure 5. Second, certain subcategory axioms are useful as subcategory definitions, but would translate to invalid class subsumptions. Some examples are shown in Fig. 5: in the lower part, *Songs* is a subcategory of *Song awards*, but the class *Song* should not be a subclass of *Award*, and both should not be subclasses of *Award Winners*.

If this step was omitted and every subcategory assertion was converted to a subclass assertion, it would lead to a massive amount of false type assertions in the final graph (in the case above: typing a song with *Award* and *Award Winner*). Such cases are quite frequent in the Wikipedia category system. For example, consider the following sequence of categories: *Electronic Rock Songs* < *Electronic Rock* < *Rock Music Genres*. If the subcategories were naively transferred to classes, each of the 82 instances of the class *Electronic Rock Song* would ultimately be additionally typed as *Genre*. The filtering of wrong category edges therefore prevents a larger number of wrong type assertions in CaLiGraph.

As a final step, we connect the taxonomy of categories and lists to the DBpedia ontology. We map the categories to DBpedia classes using type axioms derived from DBpedia resources and linguistic signals (see next section for how the type axioms are created). After mapping, the CaLiGraph ontology consists of all the information in DBpedia like classes, properties, axioms (e.g., class disjointnesses), resources, and additional classes from categories and list pages. In the following, elements of the T-box (i.e., classes and properties) of CaLiGraph will be prefixed with *clgo* while elements of the A-box (i.e., resources) will be prefixed with *clgr*. Categories and list pages are converted to CaLiGraph classes using their singular form. For example, the category *Albums* becomes *clgo:Album* and the list page *List of greatest hits albums* becomes *clgo:Greatest_hits_album*.

3.2.3. Axiom Learning

While category names are plain strings, we aim to uncover the semantics of the categories. To that end, we want to extract both type and relation assertions from categories and assign them to entities in those categories. Formally, we can learn two types of axioms:

$$\exists category. \{Category\} \sqsubseteq Class \quad (1)$$

$$\exists category. \{Category\} \sqsubseteq \exists relation. \{Entity\} \quad (2)$$

In Figure 4, we may learn the following ontology axioms:

$$\exists category. \{Nine_Inch_Nails_albums\} \sqsubseteq Album \quad (3)$$

$$\exists category. \{Nine_Inch_Nails_albums\} \sqsubseteq \exists artist. \{Nine_Inch_Nails\} \quad (4)$$

Given that we have an instance, e.g., *Pretty_Hate_Machine* in the category *Nine_Inch_Nails_albums*, we can infer two assertions for the instance, i.e.,

$$(Pretty_Hate_Machine, a, Album) \quad (5)$$

$$(Pretty_Hate_Machine, artist, Nine_Inch_Nails) \quad (6)$$

The derived type axioms serve as the basis for a mapping from categories to DBpedia. The relation axioms are added to the CaLiGraph ontology as restrictions similar to *restriction1* in Figure 1.

The *Cat2Ax* approach [48] derives such axioms by combining signals from the category graph structure, linguistic patterns in category names, and instance information from the KG. To generate axioms, the approach defines the three phases *Candidate Selection*, *Pattern Mining*, and *Pattern Application*:

Candidate Selection We identify sets of categories that most likely share a common type or relation. In Figure 1, an example of a category set is the set of subclasses of the category *Albums* by *artist*. Category sets must have a common supercategory and a shared pre- or postfix, which may serve as a linguistic pattern.

Pattern Mining We use the category sets to identify linguistic patterns as pre- or postfixes for all possible types and relations. For example, we may learn the pattern that categories ending in *albums* have a high likelihood of having entities with the type *Album* and an *artist* that is mentioned before *albums*. We learn such patterns from the category sets by combining two signals we retrieve via distant supervision over DBpedia: The first signal comes from type and relation frequencies for entities of the categories. The second signal is the information about lexicalisation frequencies (e.g., given the text *Nine Inch Nails*, how likely is it that this text refers to the entity *Nine_Inch_Nails* in DBpedia?). By combining those two independent signals, we mitigate one of the key problems of distant supervision, i.e., noisy training data, since a training example is only added if it has both a high frequency score as well as a high lexical score.

Pattern Application We apply the patterns to all categories in Wikipedia to extract axioms like (1) and (2). Here, we combine the likelihood of a pattern with the signals from a category to judge whether applying the pattern to the category is possible.

3.3. Knowledge Graph Population

This section describes the steps taken to populate CaLiGraph with additional entities as well as type and relation assertions (cf. lower part of Figure 1). We first recognize mentions of SEs in listings [45], then we link the mentions to entities in CaLiGraph or create new entities [47], and finally we derive new facts for the entities discovered in listings [46].

3.3.1. Named Entity Recognition

While a few listings already contain disambiguated entities, this is not the case everywhere. Many listings contain only text, mostly because the entities in them do not have a corresponding Wikipedia page describing them. Thus, we need an additional entity recognition step.

To detect SEs in listings, we phrase the problem as a token classification problem [45]. We produce a label for every token of the input sequence and aggregate the token labels to predictions of SE mentions. With a transformer-based model, we predict 13 token labels, such as *Person* or *Organisation*. By that, we not only identify SEs, but also get a prediction of their type used in the subsequent disambiguation step.

We pass the context of a listing (e.g., page name and section name) and multiple listing items as textual input to the transformer model. To preserve the information about context and listing layout, we use special tokens. By passing multiple listing items at once, the model can learn the structure of the listing. For example, it may recognize that the SE is always mentioned in the first cell of a table (cf. Fig. 3a and 3d), or is always followed by a particular sequence of characters (cf. Fig. 3b and 3c).

We generate the training data for the mention detection model from entities in list pages, using a heuristic labeling (i.e., weak supervision): as we already know the type of entities in a list page (e.g., entities in *List of greatest hits albums* must be of type *Album*, as it is a subclass in the CaLiGraph taxonomy), we use all entities with the matching type in a list page as positive examples. Entities with a disjoint type are regarded as negative examples.

3.3.2. Named Entity Disambiguation

One main challenge of Named Entity Disambiguation (NED) is the inherent ambiguity of mentioned entities in the text. Figure 3 shows four homonymous mentions of distinct entities with the name *James Lake* (a lake in Canada, a lake in the US, a musician, and a fictional character). Correctly linking the mentions in 3a and 3b is especially challenging as both point to geographically close lakes. In a practical setting, we additionally encounter the problem of mentions without a corresponding entity in the KG (which we refer to as NIL mentions and NIL entities, respectively). The mention in Figure 3a is the only one with a counterpart in DBpedia.

With NASTyLinker [47], we employ an approach for NED in CaLiGraph that can deal with both of these challenges. It produces clusters of mentions and entities based on inter-mention and mention-entity affinities from a bi-encoder. NASTyLinker relies on a top-down clustering approach that assigns mentions to the entity with the highest transitive affinity in case of a conflict. A threshold on the transitive affinity ensures that new entities are created for mentions without an existing counterpart in CaLiGraph.

3.3.3. Information Extraction

The information extraction efforts in CaLiGraph are currently focused on SEs in Wikipedia listings. Our approach identifies the characteristics of a listing, which are the types and relations shared by all its SEs. Given the example page about *Gilby Clarke* in the lower part of Figure 1, we want to learn that all mentioned SEs are musical works with *Gilby Clarke* as an artist. For the SEs *The Spaghetti Incident?* and *Greatest Hits*, we additionally want to learn that they have the band *Guns N' Roses* as an artist.

We frame finding descriptive rules for listings based on their context as an association rule mining problem [46]. We define rule metrics that take the inherent uncertainty into account and make sure that rules are frequent (rule support), correct (rule confidence), and consistent for all listings (rule consistency). To find a reasonable balance between the correctness and coverage of the rules, we set the thresholds based on a heuristic considering the distribution of NE tags over entities and existing knowledge in CaLiGraph. For the example given above, we identify the following generic rules:

$$\exists \text{topSection}.\{\text{"Discography"}\} \sqsubseteq \text{Musical_work} \quad (7)$$

$$\exists \text{pageEntityType}.\{\text{Person}\} \sqcap \exists \text{topSection}.\{\text{"Discography"}\} \sqsubseteq \exists \text{artist}.\{\langle \text{PageEntity} \rangle\} \quad (8)$$

$$\exists \text{topSection}.\{\text{"Albums with } \langle \text{SectionEntity} \rangle\} \sqsubseteq \exists \text{artist}.\{\langle \text{SectionEntity} \rangle\} \quad (9)$$

4. An Overview of CaLiGraph

This section gives an overview of CaLiGraph as a data source. First, we introduce its versions, purpose, and vocabulary structure. Then, we detail the extraction procedure of CaLiGraph, including sources, provenance, stability, and sustainability. Finally, we explain how CaLiGraph can be accessed and how it is used already.

4.1. Description

CaLiGraph and all the associated information is accessible via <http://caligraph.org>. The dataset is licensed under CC BY 4.0,¹⁵ giving everyone the right to use, share and adapt all material with the only liability of giving proper attribution.

The project to create CaLiGraph was initiated in 2018 [53] and, to date, three major versions have been published. Here is an overview of the versions that we use in the remainder of this work:

- **CLGv1** (version 1.1.0 from 20.09.2020): contains the full class hierarchy and axioms as described in Section 3.2.

¹⁵<https://creativecommons.org/licenses/by/4.0>

- **CLGv2** (version 2.1.1 from 21.09.2021): adds additional entities extracted from all listings in Wikipedia as described in Section 3.3.1 and additional facts from associated rules as described in Section 3.3.3. However, this version may contain duplicate entities not properly disambiguated during extraction.
- **CLGv3** (version 3.1.1 from 22.06.2023): adds an entity disambiguation step as described in Section 3.3.2.

4.1.1. Purpose and Coverage

The purpose of CaLiGraph is to serve as a large-scale general-purpose KG covering all topics addressed in Wikipedia. In particular, CaLiGraph aims to incorporate all information given in a semi-structured format in Wikipedia. By exploiting the data structure, the extraction mechanisms of CaLiGraph can extract information, especially about long-tail entities, more precisely than from full text. Currently, the focus is on extracting information about entities mentioned in tables and enumerations.

Another feature distinguishing CaLiGraph from most other public general-purpose KGs is its large taxonomy containing expressive class descriptions. An example is shown in the upper part of Figure 1 where *restriction1* enforces that all entities in the class `Guns N' Roses album` have the band `Guns N' Roses` as an artist. With such restrictions, we model the meaning of the classes that is usually hidden behind their names.

4.1.2. Vocabulary

The CaLiGraph dataset builds on well-established vocabularies like *RDFS*, *OWL*, *SKOS*, *FOAF* and *PROV* to describe important concepts like classes, hierarchies, restrictions, labels, and provenance. Overall, the complexity of the CaLiGraph ontology can be categorised as *SHOD* [13]. From an ontological perspective, the feature distinguishing CaLiGraph most from other public general-purpose KGs is its extensive modelling of class restrictions via *OWL* as described in the previous section [54].

4.2. Extraction Procedure

CaLiGraph is extracted using the CaLiGraph Extraction Framework¹⁶ as described in Section 3. We describe the extraction's inputs, outputs, and organisation in the following.

4.2.1. Data Sources

The main inputs to the CaLiGraph extraction framework are an XML dump of the English Wikipedia and the English chapter of DBpedia [18] in the form of triples. Further, we use WebIsALOD [52] to gather additional hypernyms during taxonomy construction (see Section 3.2.2).

4.2.2. Provenance

In CaLiGraph, we provide provenance information for new classes and entities using *PROV* vocabulary. For existing classes, properties and entities taken from DBpedia, we add links via *rdfs:subClassOf*, *owl:equivalentProperty* and *owl:sameAs*, respectively. Similar to DBpedia, we use the namespaces `http://caligraph.org/ontology/` or short *clgo* for the ontology and `http://caligraph.org/resource/` or short *clgr* for resources.

For additional classes, we point to the Wikipedia categories or list pages used for extraction. For the additional entities, we include information about the listings they have been extracted from. For example, suppose we create the new class `Lake of Powell County, Montana` from the list page `List of lakes of Powell County, Montana`. In that case, we add the following triple for provenance:¹⁷

```
clgo:Lake_of_Powell_County,_Montana prov:wasDerivedFrom wiki>List_of_lakes_of_Powell_County,_Montana .
```

As shown above, when minting new classes or entities, we again follow DBpedia and create a URI similar to its main label. We use the source's name as a prefix in case of name clashes.

¹⁶<https://github.com/nheist/CaLiGraph>

¹⁷We use the *prov* namespace for *PROV* vocabulary and the *wiki* namespace to refer to Wikipedia pages.

4.2.3. Stability

CaLiGraph is built on information from Wikipedia and DBpedia. New releases are dependent on the information from these two resources. As we have no control over the data sources, CaLiGraph gives no guarantees for the stability of ontology and resources between major versions. The changes may affect any information contained in CaLiGraph. For example, it is possible that a page name in Wikipedia and, consequently, a resource in DBpedia changes. This change would then be taken over in CaLiGraph as well. Further, if the structure of the category graph in Wikipedia changes, this can influence the extraction of the CaLiGraph taxonomy. Finally, any changes in listings in Wikipedia may change how facts are extracted.

4.2.4. Sustainability

CaLiGraph is hosted and maintained by the Data and Web Science Group of the University of Mannheim.¹⁸ The release cycle for CaLiGraph was mostly irregular in the past, as new developments were integrated as quickly as possible. There are ongoing efforts to align the release cycle to the one of DBpedia and even to integrate the extraction of CaLiGraph into the DBpedia extraction workflow. Still, it is planned to improve and extend CaLiGraph further in various ways (see future work in Section 6.3). The Data and Web Science group plans for several projects for improving the quality of CaLiGraph, each of which will be accompanied by a new full extraction. Like for other efforts carried out by the group, such as WebDataCommons,¹⁹ we are committed to maintaining access to open datasets, as well as to further develop the datasets.

4.3. Usage

The following describes how to access and interact with CaLiGraph best. Further, we give an overview of potential and existing use cases.

4.3.1. Access

The main web resources to view, use, and extend CaLiGraph are:

- <http://caligraph.org>: Main website with relevant resources, a data explorer and a SPARQL endpoint.
- <https://zenodo.org/record/3484511>: Source files of all published versions of CaLiGraph.
- <https://databus.dbpedia.org/nheist/CaLiGraph>: CaLiGraph on the DBpedia Databus.
- <https://github.com/nheist/CaLiGraph>: Code of the extraction framework, including an issue tracker.

4.3.2. Use Cases

In general, CaLiGraph is intended to be used as a knowledge base for various domains similar to DBpedia. Hence, it can be used in similar use cases, for example, for information retrieval or question answering. CaLiGraph is already used in several concrete scenarios:

- Qin and Iwaihara [55] use CaLiGraph as training data for a transformer model to annotate table columns with entity types.
- Biswas et al. [56] use CaLiGraph to evaluate models for entity typing using only the surface forms of the entities.
- In 2021, we submitted CaLiGraph as a dataset for the Semantic Reasoning Evaluation Challenge [54]. It has been used in every challenge edition to evaluate reasoning systems (for example, by Chowdhury et al. [57]).

5. Statistics, Quality, and Evaluation

In this section, we show statistics about CaLiGraph, summarize all efforts to measure its quality and compare its performance on downstream tasks with DBpedia and YAGO. We use the English chapters of DBpedia in the versions from 2016 (*DBP16*) and 2022 (*DBP22*) as well as YAGO version 3.1 (*YAGO3*). We select these KGs for comparison as they are, like CaLiGraph, mainly based on the English Wikipedia.

¹⁸<https://www.uni-mannheim.de/dws/>

¹⁹<https://webdatacommons.org/>

Table 2

Basic metrics of all CaLiGraph versions and other KGs based on the English Wikipedia. †Entities are not disambiguated properly.

	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
# Classes	760	1,245	819,292	755,440	1,061,597	1,285,484
# Relations	1,355	1,298	77	271	343	1,253
# HasValue Restrictions	–	–	–	110,180	128,016	145,631
Avg. depth of class tree	3.5	3.9	6.6	4.5	3.9	3.6
Avg. branching factor of class tree	4.5	4.2	8.5	4.5	4.4	4.9
Ontology complexity	SHOFD	SHOFD	SHOIF	SHOD	SHOD	SHOD
# Entities	5,044,223	7,495,054	6,349,359	6,516,892	15,230,974†	13,736,724
# Assertions	71,628,627	97,213,941	263,433,367	166,228,505	298,300,766	332,884,815
Avg. linking degree	2.8	2.7	1.9	1.6	0.7	1.7
Median ingoing edges	0	1	0	0	0	0
Median outgoing edges	15	11	35	17	12	12

5.1. Statistics

We compare the KGs w.r.t. classes and entities in Table 2. We have performed a similar comparison with more public KGs in a previous work [13], but only with an early version 1.0.6 of CaLiGraph. However, the results are not directly comparable as, in the previous study, we only considered predicates in the namespace of the respective KG.

Compared to DBpedia, YAGO and CaLiGraph contain many more classes, largely retrieved from the WCG. The increase in classes and relations in the major CaLiGraph versions is caused by the Wikipedia version used for extraction (*CLGv1* uses a version from 2016, *CLGv2* from 2020 and *CLGv3* from 2022). *YAGO3* uses a Wikipedia version from 2017; an extraction on a recent version would likely increase the number of classes. Regarding the class tree’s depth and branching factor, DBpedia and CaLiGraph are comparable, while YAGO has a denser and deeper taxonomy.

In terms of entities, *CLGv2* and *CLGv3* contain the highest number, almost twice as many as the other KGs. While the entities in *CLGv2* may not be properly disambiguated, *CLGv3* employs a disambiguation mechanism to ensure that no duplicate entities exist (see Section 3.3.2). *CLGv3* contains the highest number of assertions about entities, but the average number of assertions per entity is higher in *YAGO3*. This can be attributed mostly to literal assertions of YAGO, as the median of outgoing edges is high, but the average linking degree is comparably low. Compared to YAGO and CaLiGraph, DBpedia is interlinked very strongly, indicated by the high average linking degree.

Similarly to DBpedia and YAGO, CaLiGraph covers many domains. Figure 6 shows how the entities in *CLGv3* are distributed over the type hierarchy. Most entities describe *Species*, a majority thereof *Persons*.²⁰ Next in line are *Works*, most importantly musical works and movies. Entities describing *Places* are mostly addressing *Populated places*. The large number of *Places in Myanmar* can be traced back to an incorrect mapping in the taxonomy, which will be fixed in the next release (see Section 6.2 for more details). The large class *Birth* is derived from the corresponding Wikipedia category,²¹ which organizes persons by their year of birth. This is also an example where the identification of incorrect class subsumptions among subcategory definitions leads to a false removal: since *birth* and *people* (which is the supercategory in the Wikipedia category graph) are neither synonyms nor hyponyms, the subclass relation between *Birth* and *People* (or *Person*) does not exist in CaLiGraph.

We compare the type and relation frequencies of the three CaLiGraph versions in Table 3. We use the prominent types mentioned in Heist et al. [13] to compare types. Unfortunately, the ranks are not perfectly comparable as DBpedia changed its taxonomy, taking effect in *CLGv3*. As a consequence, *Person* is a descendant of *Species* instead of *Agent*. This explains why *Species* is the most frequent type in *CLGv3* and why *Person* descended to rank five while almost doubling its numbers compared to *CLGv1*. While the coverage of organizations and places has

²⁰Due to the fact that in the DBpedia ontology, *Person* is an indirect subclass of *Species*.

²¹<https://en.wikipedia.org/wiki/Category:Births>



Fig. 6. A sunburst diagram of frequent entity types in CaLiGraph.

not increased much between versions 1 and 3, the counts of *Work* (+200%), *Building_or_structure* (+160%), *Gene* (+1,160%) and *Event* (+700%) multiplied.

We take the most frequently used ones in *CLGv3* to compare properties. Again, some changes can be explained with the changes in DBpedia, like the increased coverage of *birthYear* (about 1K in *DBP16* and 260K in *DBP22*) and the added support for *subdivision* and *birthDate*. The decline in rank for *country* aligns with the observations on types, indicating that locations are already well covered in DBpedia.

Table 3

Comparison of counts and ranks of prominent types and properties among CaLiGraph versions. Prominent types are taken from Heist et al. [13], and prominent properties are selected based on their frequency in CLGv3.

	CLGv1		CLGv2		CLGv3	
Types	Count	Rank	Count	Rank	Count	Rank
Person	1,827,240	2	4,599,249	2	3,262,511	5
Organization	593,462	13	1,106,098	19	691,732	25
Populated_place	648,673	9	1,014,971	24	867,281	20
Natural_place	132,618	102	180,930	125	164,987	102
Species	353,680	26	790,287	30	3,691,343	1
Work	607,858	12	2,468,257	7	1,832,985	8
Building_or_structure	202,888	67	619,983	35	524,310	37
Gene	1,112	9,149	25,852	817	14,019	1,269
Protein	6,049	1,882	3,882	4,455	5,264	3,138
Event	141,582	90	309,674	67	1,178,248	14
Properties	Count	Rank	Count	Rank	Count	Rank
birthPlace	2,827,536	1	2,844,951	1	3,887,146	1
birthYear	1,128	133	1,175,897	4	1,576,909	2
location	877,066	3	1,485,013	2	1,567,334	3
team	521,660	5	748,147	6	1,338,344	4
country	963,588	2	1,265,201	3	1,315,784	5
subdivision	-	-	844,344	5	1,233,513	6
birthDate	-	-	-	-	976,431	7
type	284,581	8	390,191	8	716,693	8
genre	326,955	7	405,484	7	711,336	9
deathYear	111,273	15	11,916	78	706,773	10

5.2. Data Quality

We provide information about the data quality in CaLiGraph concerning its metadata (Section 5.2.1), the vocabulary use (Section 5.2.2), as well as class and instance data (Section 5.2.3).

5.2.1. Metadata

As described in Section 4, the CaLiGraph ontology builds on well-established vocabularies like *SKOS*, *FOAF* and *PROV*. Further, the KG is described in various aspects (e.g., purpose, creators, version) with the vocabulary of the *Dublin Core*. In 2023, Andersen et al. [58] conducted an experiment evaluating the accountability of 670 KGs in the LOD cloud. More concretely, they evaluated how much information KGs provide about their data collection, maintenance and usage (e.g., who created the KG and how was it created?). They retrieve this information via SPARQL queries. Of the 670 KGs, only 29 responded to queries; of those, CaLiGraph was ranked fifth. Based on the results of the experiments, we added more metadata so that, all else equal, CaLiGraph would now be ranked first.

5.2.2. Five Star Rating

According to the five-star rating for linked data vocabulary use defined by Janowicz et al. [59], the CaLiGraph dataset can be categorized as a four-star dataset and will be a five-star dataset soon:

1. Star: There is dereferenceable human-readable information about the used vocabulary on <http://caligraph.org>.
2. Star: The information is available as machine-readable explicit axiomatization of the vocabulary as the CaLiGraph ontology is published using *RDFS* and *OWL*.
3. Star: The vocabulary is linked to other vocabularies, e.g., *DBpedia* (see Section 4.2.2).
4. Star: Metadata about the vocabulary is available (see Section 5.2.1).

Table 4
Collection of evaluation results of CaLiGraph data.

Target	Method	Metric	#Samples	Result	Source
Edges in the taxonomy (cf. Section 3.2.2)	Manual evaluation via Amazon MTurk ²⁴ (majority of three votes)	Accuracy	2,000	96.25% (±0.86%)	[49]
Type restrictions (cf. Section 3.2.3)	Manual evaluation via Amazon MTurk (majority of three votes)	Accuracy	250	96.8%	[48]
Relation restrictions (cf. Section 3.2.3)	Manual evaluation via Amazon MTurk (majority of three votes)	Accuracy	250	95.6%	[48]
Subject entities in arbitrary listings (cf. Section 3.3.1)	Evaluation on manually labelled dataset	F1-score (<i>Exact match</i>)	9,400	74%	[45]
Single mention assigned to an entity (cf. Section 3.3.2)	Manual evaluation by authors	Accuracy	100	89.4% (±9.6%)	[47]
All mentions assigned to an entity (cf. Section 3.3.2)	Manual evaluation by authors	Accuracy	100	82.3% (±7.0%)	[47]
Entity types derived from listings (cf. Section 3.3.3)	Manual evaluation by authors	Accuracy	2,000	91.95% (±1.19%)	[46]
Relations derived from listings (cf. Section 3.3.3)	Manual evaluation by authors	Accuracy	1,000	95.90% (±1.23%)	[46]

5. Star: The vocabulary is linked to by other vocabularies soon, as DBpedia is preparing to provide backlinks to CaLiGraph similar to the ones from CaLiGraph to DBpedia.²²

5.2.3. Class and Instance Data

In Table 4, we collect all evaluation results of parts of CaLiGraph data conducted using direct or indirect human supervision. CaLiGraph intends to ingest as much of the semi-structured information in Wikipedia as possible. The results show that most of the information is extracted with an accuracy of over 90%, with entity linking approaches being the only exception.

One metric that allows for direct comparison with other knowledge graphs is the accuracy of relation assertions, which essentially refers to the fraction of correct triples. Here, CaLiGraph yields an accuracy of about 96%, which is slightly below the reported triple accuracy of DBpedia, YAGO, and Wikidata, which, according to [60], expose a triple accuracy (called *semantic validity* in their paper) of 99%.²³

The CaLiGraph extraction pipeline is a sequence of steps, with later ones depending on the results of previous steps. It is, hence, unavoidable that errors are propagated through the pipeline. The evaluations listed in Table 4 identify such errors explicitly. In the results of NASTyLinker [47], the errors are not contained in the final accuracy of 89.4% for single mentions and 82.3% for all mentions. Considering the SE labeling errors [45], the results for single mentions would decrease by 5.4%, and the results for all mentions would decrease by 3.3%. The results for extracting facts from listings [46] include errors caused by incorrectly parsed entities already. The errors are responsible for an accuracy decrease of 2.6% for entity types and 0.2% for relations.

The named entity disambiguation has an important impact on the overall quality of CaLiGraph. In [47] we explored that impact by comparing it to simple baselines. The most trivial of entity disambiguation baselines, i.e., assuming that all SEs with the same label denote the same entity, leads to both lower precision (91.4% compared to 97%) and recall (73.5% compared to 87.0%) than the named entity disambiguation used in CaLiGraph. The drop in recall is more significant since it is more likely that an entity has multiple labels than that the same label refers to multiple entities.

Moreover, only by clustering entity mentions and allowing for NIL entities, it is possible to attribute information to entities which do not have their own Wikipedia page. This is the most remarkable difference of CaLiGraph compared to other knowledge graphs derived from Wikipedia, which only contain entities for which a dedicated Wikipedia page exists.

²²<https://www.dbpedia.org/resources/latest-core/>

²³The direct comparison of the numbers, though, has to be taken with a grain of salt: while our evaluation is based on a random sample across all extracted triples, the evaluation in [60] is based solely on four properties of the class *Person*.

5.3. Evaluation via Downstream Tasks

KGrEaT (**K**nowledge **G**raph **E**valuation via **D**ownstream **T**asks) [61] aims to provide a comprehensive assessment of KGs through evaluation on multiple kinds of tasks like classification, regression, or recommendation. The evaluation results (e.g., the accuracy of a classification model trained with the KG as background knowledge) serve as extrinsic task-based quality metrics for the KG. By defining a fixed evaluation set up in the framework and applying it to multiple KGs, it is possible to isolate the effect of every KG and compare their usefulness in solving different tasks. In the following, we use KGrEaT to assess the utility of CaLiGraph and compare it to related KGs.

5.3.1. Experimental Setup

We consider CaLiGraph (*CLGv1*, *CLGv2*, *CLGv3*), DBpedia (*DBP16*, *DBP22*) and *YAGO3* in our comparison. We run the evaluation for all seven tasks in KGrEaT: Classification, Regression, Clustering, Document Similarity, Entity Relatedness, Semantic Analogies and Recommendation. The tasks are evaluated on 20 datasets covering areas like geography, music, movies or literature. *MillionSongDataset*, *ComicCharacters*, *MovieLens*, *LibraryThing* and *LastFm* are datasets derived from independent sources; the remaining datasets are created from DBpedia version 2015.

We report the results for two entity mapping scenarios: precision-oriented mapping and recall-oriented mapping. Both scenarios link the task dataset’s entities to KG entities using `owl:sameAs` links and labels. The former scenario uses a precision-focused label mapper, while the latter uses a label mapper focused on recall [61]. In the precision-oriented scenario, we consider only mapped entities in the evaluation, while in the recall-oriented scenario, we consider all entities.

We compute the results using four embedding methods: *TransE* [62], *DistMult* [63], *Complex* [64], and *RDF2vec* [65]. We run evaluations with embeddings trained for one and two epochs, respectively. In total, we compute results for eight configurations for every KG and scenario; we take the best approach w.r.t. embedding and algorithm, and then we aggregate the results by task, dataset and metric.

5.3.2. Results and Discussion

Table 5 shows the average rank of the KGs w.r.t. the datasets of a task. In both scenarios, DBpedia shows superior performance in the Clustering, Entity Relatedness, and Semantic Analogies tasks, *YAGO* works best for Document Similarity, and CaLiGraph for Regression and Recommendation. While DBpedia has a tendency to work better in the precision-oriented scenario, CaLiGraph works better in the recall-oriented scenario.

On a dataset level (see Tables 7 and 8 in Appendix B for details), it becomes clear that the choice of a KG for a given task is always dependent on the domain. As expected, DBpedia performs well on DBpedia-based datasets. The superior performance of *DBP16* compared to *DBP22* may be explained by the temporal proximity of *DBP16* to DBpedia version 2015, serving as the source for many datasets. For almost all independent datasets, we find that CaLiGraph and *YAGO* have much higher coverage than DBpedia (see Table 6 in Appendix B for details). Especially *CLGv3* shows the highest coverage for all these datasets in the recall-oriented scenario. Consequently, using CaLiGraph for *ComicCharacters* and *MillionSongDataset* (used in Classification and Clustering) as well as for *MovieLens* and *LibraryThing* (used in Recommendation) produces superior results. Against our expectations, however, DBpedia shows a competitive performance for the independent datasets of the Recommendation task.

6. Conclusion and Outlook

6.1. Summary

With CaLiGraph, we presented a KG created from Wikipedia categories and lists, offering a rich taxonomy with semantic class descriptions, and going beyond the one-entity-per-page paradigm of DBpedia and *YAGO*, thus offering a much larger set of entities. We gave an overview of its extraction framework and summarized relevant information for potential users of the KG. The comparison of CaLiGraph to other popular public KGs shows that, despite its wealth in classes and entities, it can be favorable to use CaLiGraph in some scenarios, but there is no one-size-fits-all solution.

Table 5

Evaluation results of the KGs given as average rank per task type. The results are computed for a precision-oriented mapping scenario and a recall-oriented mapping scenario. The best results are bold, second-best are underlined.

Task Type	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
<i>Precision-oriented mapping</i>						
Classification	1.8	<u>2.2</u>	3.5	5.0	5.3	3.2
Regression	<u>3.0</u>	3.2	3.6	4.0	5.6	1.6
Clustering	2.3	<u>2.8</u>	<u>2.8</u>	4.4	4.4	4.3
Document Similarity	4.0	6.0	1.3	3.0	<u>1.7</u>	5.0
Entity Relatedness	<u>2.0</u>	1.0	4.0	5.0	6.0	3.0
Semantic Analogies	1.8	3.5	4.0	3.8	6.0	<u>2.0</u>
Recommendation	<u>2.7</u>	3.3	4.3	3.0	5.3	2.3
<i>Recall-oriented mapping</i>						
Classification	3.3	5.1	3.4	2.3	4.4	<u>2.4</u>
Regression	5.6	5.4	3.0	<u>2.2</u>	3.6	1.2
Clustering	2.5	4.5	<u>3.0</u>	3.5	3.9	3.7
Document Similarity	4.0	6.0	1.0	<u>2.3</u>	2.7	5.0
Entity Relatedness	1.0	3.0	4.0	5.0	6.0	<u>2.0</u>
Semantic Analogies	1.8	3.8	4.0	3.0	6.0	<u>2.5</u>
Recommendation	<u>2.0</u>	3.3	5.0	4.0	5.0	1.7

6.2. Limitations

In Section 2.3, we identified several challenges in the field of AKGC. We made a step forward for some of them, while others are yet to be addressed. For CaLiGraph, we can formulate some of these limitations in more detail:

Error Accumulation. AKGC in CaLiGraph is executed as a pipeline of automatic processing steps. Errors in early steps are propagated to subsequent steps and may create distortions with a high impact on the outcome. For example, in the recent version of CaLiGraph, an extraction error made *Place in Myanmar* a superclass of *Village* (explaining its large proportion in Figure 6). As this error occurred during OC, it affects all steps of KGP.

Entity Ambiguity. Ambiguity is one of the biggest challenges when identifying and disambiguating mentions of entities in text. As information about long-tail entities during extraction is sparse, the quality of such entities in CaLiGraph is not satisfactory yet.

Wikipedia Dependency. Currently, the CaLiGraph extraction targets a single version of Wikipedia only. Any information not contained in that version can consequently not be part of the KG. Further, we have no direct influence on the content of Wikipedia and hence have to deal with potential problems only during extraction.

DBpedia Dependency. CaLiGraph builds on the ontology of DBpedia, taking over all types and properties. While types are extended, the set of properties remains fixed, and knowledge can only be modelled within the bounds defined by the DBpedia ontology.

6.3. Future Work

For future work in CaLiGraph, the focus is divided between improving the quality of the existing KG and extending its coverage to incorporate more knowledge.

Improving Extraction Quality. While error propagation is currently problematic in CaLiGraph, it is also a chance to improve the overall quality of the graph by gradually improving the individual parts. Fixing errors in the early stages of the extraction may positively influence the complete extraction pipeline. To that end, we plan to implement a more rigorous error-monitoring system to capture errors early and monitor all parts of the pipeline to identify opportunities for improvement.

As a concrete improvement, we plan to replace or augment the taxonomy induction step of Section 3.2.2 with a Transformer model that is tuned on identifying subclass relationships (e.g., from Hertling and Paulheim [66]). This may improve the class hierarchy substantially as we currently rely on manually combined hypertext information from multiple sources.

We plan to put more emphasis on the dependency of SEs expressed through co-occurrence. This might be particularly helpful when trying to disambiguate entities in text. We are only implicitly using the context of an entity mention during disambiguation. Explicitly providing information about related entities might improve the disambiguation capabilities of NASTyLinker.

Extending KG Coverage. We plan to extend CaLiGraph in the three dimensions of ontology, assertions and data sources. To extend the ontology, we can discover additional axioms by extending the Cat2Ax approach from categories to list pages. Additionally, we may derive more axioms by relying on common sense knowledge from another KG (e.g., CSKG [67]). We further plan to discover new properties by using the existing data in CaLiGraph as a foundation to automatically exploit dependencies between co-occurring entities where the relation underlying the co-occurrence pattern is not in the ontology yet.

Like YAGO, we can extend the coverage of CaLiGraph to more dimensions like temporal or geospatial information. As the KG currently reflects only the point in time when the Wikipedia dump was created, we consider incorporating edits in Wikipedia pages to reflect the temporal dimension. Alternatively, we explore the possibility of extracting CaLiGraph from multiple dumps and merging the results to include a temporal perspective.

CaLiGraph currently targets only the English Wikipedia chapter. An extension to other languages would have the benefit of providing multilingual labels. Still, all the automatic extraction mechanisms may be able to derive much more complementary information from the diverse language chapters. The main challenge here is to merge the information derived from all the language chapters into a unified KG. Finally, we may extend the extraction to other data sources. As most extraction methods in the pipeline are built for encyclopedic content, a first step is to follow the example of DBkWik and target other Wikis than Wikipedia.

Acknowledgement

The publication of this article was funded by the University of Mannheim.

Appendix A. Data Sources for Knowledge Graph Comparison and Evaluation

For the comparison and evaluation of the KGs, we used the following data:

A.1. CaLiGraph

Version 1.1.0 (based on Wikipedia from 2016)

- <https://zenodo.org/record/4050308/files/caligraph-ontology.nt.bz2>
- https://zenodo.org/record/4050308/files/caligraph-ontology_dbpedia-mapping.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-ontology_provenance.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_types.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_transitive-types.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_labels.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_relations.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_dbpedia-mapping.nt.bz2
- https://zenodo.org/record/4050308/files/caligraph-instances_provenance.nt.bz2

Version 2.1.1 (based on Wikipedia from 2020)

- 1 – <https://zenodo.org/record/5524052/files/caligraph-ontology.nt.bz2> 1
- 2 – https://zenodo.org/record/5524052/files/caligraph-ontology_dbpedia-mapping.nt.bz2 2
- 3 – https://zenodo.org/record/5524052/files/caligraph-ontology_provenance.nt.bz2 3
- 4 – https://zenodo.org/record/5524052/files/caligraph-instances_types.nt.bz2 4
- 5 – https://zenodo.org/record/5524052/files/caligraph-instances_transitive-types.nt.bz2 5
- 6 – https://zenodo.org/record/5524052/files/caligraph-instances_labels.nt.bz2 6
- 7 – https://zenodo.org/record/5524052/files/caligraph-instances_relations.nt.bz2 7
- 8 – https://zenodo.org/record/5524052/files/caligraph-instances_dbpedia-mapping.nt.bz2 8
- 9 – https://zenodo.org/record/5524052/files/caligraph-instances_provenance.nt.bz2 9

Version 3.1.1 (based on Wikipedia from 2022)

- 11 – <https://zenodo.org/record/8068322/files/caligraph-ontology.nt.bz2> 11
- 12 – https://zenodo.org/record/8068322/files/caligraph-ontology_dbpedia-mapping.nt.bz2 12
- 13 – https://zenodo.org/record/8068322/files/caligraph-ontology_provenance.nt.bz2 13
- 14 – https://zenodo.org/record/8068322/files/caligraph-instances_types.nt.bz2 14
- 15 – https://zenodo.org/record/8068322/files/caligraph-instances_transitive-types.nt.bz2 15
- 16 – https://zenodo.org/record/8068322/files/caligraph-instances_labels.nt.bz2 16
- 17 – https://zenodo.org/record/8068322/files/caligraph-instances_relations.nt.bz2 17
- 18 – https://zenodo.org/record/8068322/files/caligraph-instances_dbpedia-mapping.nt.bz2 18
- 19 – https://zenodo.org/record/8068322/files/caligraph-instances_provenance.nt.bz2 19

A.2. DBpedia

Version 2016-10 (English Chapter)

- 24 – http://downloads.dbpedia.org/2016-10/dbpedia_2016-10.nt 24
- 25 – http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_en.ttl.bz2 25
- 26 – http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_transitive_en.ttl.bz2 26
- 27 – http://downloads.dbpedia.org/2016-10/core-i18n/en/labels_en.ttl.bz2 27
- 28 – http://downloads.dbpedia.org/2016-10/core-i18n/en/mappingbased_literals_en.ttl.bz2 28
- 29 – http://downloads.dbpedia.org/2016-10/core-i18n/en/mappingbased_objects_en.ttl.bz2 29

Version 2022-09 (English Chapter)

- 31 – https://databus.dbpedia.org/ontologies/dbpedia.org/ontology/2022.09.02-100003/ontology_type=parsed.nt 31
- 32 – https://downloads.dbpedia.org/repo/dbpedia/mappings/instance-types/2022.09.01/instance-types_lang=en_specific.ttl.bz2 32
- 33 – https://downloads.dbpedia.org/repo/dbpedia/mappings/instance-types/2022.09.01/instance-types_lang=en_transitive.ttl.bz2 33
- 34 – https://downloads.dbpedia.org/repo/dbpedia/generic/labels/2022.09.01/labels_lang=en.ttl.bz2 34
- 35 – https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-literals/2022.09.01/mappingbased-literals_lang=en.ttl.bz2 35
- 36 – https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-objects/2022.09.01/mappingbased-objects_lang=en.ttl.bz2 36
- 37 – https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-objects/2022.09.01/mappingbased-objects_lang=en.ttl.bz2 37
- 38 – https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-objects/2022.09.01/mappingbased-objects_lang=en.ttl.bz2 38

A.3. YAGO

Version 3.1

- 43 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTransitiveType.ttl.7z> 43
- 44 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoSchema.ttl.7z> 44
- 45 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTypes.ttl.7z> 45
- 46 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTaxonomy.ttl.7z> 46
- 47 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoLiteralFacts.ttl.7z> 47
- 48 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoLabels.ttl.7z> 48
- 49 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoDateFacts.ttl.7z> 49
- 50 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoFacts.ttl.7z> 50
- 51 – <http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoDBpediaInstances.ttl.7z> 51

Table 6

Dataset coverage (per cent) of the KGs evaluated with KGrEaT for the precision- and recall-oriented mapping scenarios. Datasets marked with a dagger are independent of DBpedia.

Dataset	DBP16		DBP22		YAGO3		CLGv1		CLGv2		CLGv3	
	P	R	P	R	P	R	P	R	P	R	P	R
Cities	97	96	87	88	96	100	95	100	97	100	93	100
Forbes	87	87	81	81	99	100	92	100	97	100	91	100
AAUP	99	98	88	88	99	100	95	100	99	100	94	99
MetacriticMovies	98	98	95	94	90	100	100	100	100	100	98	100
MetacriticAlbums	99	99	97	97	95	100	97	100	99	100	96	100
MillionSongDataset†	6	22	6	21	11	51	10	51	20	60	20	64
Teams	100	100	94	94	77	83	99	100	95	97	94	95
ComicCharacters†	0	22	0	17	0	59	0	53	0	57	0	62
CitiesAndCountries	100	100	95	95	96	100	100	100	97	100	96	100
Cities2000AndCountries	100	100	93	93	94	99	100	100	96	100	94	100
CitiesMoviesAlbumsCompaniesUni	90	88	85	85	94	100	96	99	96	93	88	99
LP50	90	90	88	88	83	99	92	100	89	100	92	100
KORE	100	100	100	100	100	100	100	100	100	100	100	100
CurrencyEntities	100	100	93	93	100	100	100	100	97	100	97	100
CityStateEntities	96	97	97	97	99	100	98	100	97	100	82	100
CapitalCountryEntities	100	100	100	100	100	100	100	100	100	100	100	100
AllCapitalCountryEntities	99	99	96	97	99	100	98	100	97	99	96	99
MovieLens†	16	62	15	60	14	92	16	95	16	95	15	96
LibraryThing†	18	42	18	41	21	84	22	86	27	91	28	92
LastFm†	94	94	91	92	94	99	94	99	94	99	93	99

Appendix B. KGrEaT Evaluation Results

In this section, we provide additional details for the evaluation with KGrEaT. Table 6 describes the coverage of the KGs in the experiment w.r.t. the individual datasets. Tables 7 and 8 give the detailed performance numbers per dataset for the precision- and recall-oriented scenario, respectively.

References

- [1] D. Lenat and E. Feigenbaum, On the thresholds of knowledge, *Artificial Intelligence: Critical Concepts* **2** (2000), 298, [https://doi.org/10.1016/0004-3702\(91\)90055-O](https://doi.org/10.1016/0004-3702(91)90055-O).
- [2] G. Weikum, Knowledge graphs 2021: A data odyssey, *Proceedings of the VLDB Endowment* **14**(12) (2021), 3233–3238, <https://doi.org/10.14778/3476311.3476393>.
- [3] C. Gutiérrez and J.F. Sequeda, Knowledge graphs, *Communications of the ACM* **64**(3) (2021), 96–104, <https://doi.org/10.1145/3418294>.
- [4] M.A.N. Pour et al., Results of the Ontology Alignment Evaluation Initiative 2022, in: *Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21th International Semantic Web Conference (ISWC 2022), Hangzhou, China, held as a virtual conference, October 23, 2022*, CEUR Workshop Proceedings, Vol. 3324, CEUR-WS.org, 2022, pp. 84–128.
- [5] Y. Chen, J. Kuang, D. Cheng, J. Zheng, M. Gao and A. Zhou, AgriKG: an agricultural knowledge graph and its applications, in: *Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24*, Springer, 2019, pp. 533–537, https://doi.org/10.1007/978-3-030-18590-9_81.
- [6] G. Buchgeher, D. Gabauer, J. Martinez-Gil and L. Ehrlinger, Knowledge graphs in manufacturing and production: A systematic literature review, *IEEE Access* **9** (2021), 55537–55554, <https://doi.org/10.1109/ACCESS.2021.3070395>.
- [7] M. Kejriwal, Knowledge Graphs: Constructing, Completing, and Effectively Applying Knowledge Graphs in Tourism, in: *Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications*, Springer, 2022, pp. 423–449, https://doi.org/10.1007/978-3-030-88389-8_20.
- [8] Q. He, B.-C. Chen and D. Agarwal, Building The LinkedIn Knowledge Graph, 2016 (accessed Nov 15, 2023). <https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph>.
- [9] X. Huang, J. Zhang, D. Li and P. Li, Knowledge graph embedding based question answering, in: *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 105–113, <https://doi.org/10.1145/3289600.3290956>.

Table 7

KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *precision-oriented* mapping scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
Classification	Cities	Accuracy ↑	0.801	0.806	0.768	0.764	0.656	0.787
	Forbes	Accuracy ↑	0.617	0.599	0.604	0.580	0.556	0.584
	AAUP	Accuracy ↑	0.633	0.668	0.630	0.587	0.554	0.644
	MetacriticMovies	Accuracy ↑	0.739	0.746	0.764	0.720	0.712	0.723
	MetacriticAlbums	Accuracy ↑	0.764	0.660	0.654	0.637	0.613	0.658
	ComicCharacters	Accuracy ↑	–	–	–	–	–	–
	MillionSongDataset	Accuracy ↑	0.635	0.616	0.606	0.614	0.629	0.617
Regression	Cities	RMSE ↓	0.495	0.500	0.545	0.535	0.606	0.511
	Forbes	RMSE ↓	0.582	0.587	0.582	0.596	0.605	0.576
	AAUP	RMSE ↓	0.576	0.528	0.571	0.580	0.591	0.524
	MetacriticMovies	RMSE ↓	0.469	0.467	0.465	0.460	0.466	0.459
	MetacriticAlbums	RMSE ↓	0.462	0.533	0.538	0.534	0.549	0.514
	Teams	Accuracy ↑	0.996	0.999	0.994	0.995	0.997	0.998
Clustering		ARI ↑	0.259	0.333	0.063	0.052	0.039	0.249
		NMI ↑	0.215	0.285	0.056	0.042	0.030	0.211
	ComicCharacters	Accuracy ↑	0.667	0.667	0.875	0.800	0.800	0.875
		ARI ↑	0.000	0.000	0.495	0.231	0.000	0.505
		NMI ↑	0.734	0.734	0.562	0.380	0.101	0.529
	CitiesAndCountries	Accuracy ↑	0.935	0.940	0.982	0.790	0.898	0.800
		ARI ↑	0.740	0.755	0.916	0.071	0.614	0.037
		NMI ↑	0.657	0.658	0.831	0.205	0.545	0.203
	Cities2000AndCountries	Accuracy ↑	0.975	0.972	0.962	0.956	0.972	0.956
		ARI ↑	0.902	0.891	0.854	0.833	0.892	0.830
		NMI ↑	0.841	0.818	0.777	0.745	0.818	0.744
	CitiesMoviesAlbums-	Accuracy ↑	0.994	0.970	0.990	0.982	0.932	0.959
	CompaniesUni	ARI ↑	0.988	0.942	0.976	0.958	0.892	0.893
		NMI ↑	0.975	0.915	0.958	0.933	0.867	0.887
	Document Similarity	LP50	Spearman ↑	0.412	0.342	0.447	0.433	0.440
		Pearson ↑	0.631	0.596	0.658	0.655	0.660	0.613
		Harm. Mean ↑	0.492	0.431	0.532	0.521	0.528	0.472
Entity Relatedness	KORE	Kendall's Tau ↑	0.440	0.441	0.299	0.225	0.197	0.389
Semantic Analogies	CurrencyEntities	Accuracy ↑	0.368	0.356	0.465	0.037	0.031	0.182
	CityStateEntities	Accuracy ↑	0.500	0.453	0.212	0.480	0.072	0.721
	CapitalCountryEntities	Accuracy ↑	0.958	0.881	0.798	0.919	0.435	0.923
	AllCapitalCountryEntities	Accuracy ↑	0.911	0.884	0.697	0.880	0.643	0.929
Recommendation	MovieLens	F1-score ↑	0.013	0.019	0.022	0.023	0.022	0.033
	LibraryThing	F1-score ↑	0.047	0.031	0.016	0.021	0.012	0.019
	LastFm	F1-score ↑	0.050	0.040	0.022	0.029	0.021	0.042

- [10] X. Wang, T. Huang, D. Wang, Y. Yuan, Z. Liu, X. He and T.-S. Chua, Learning intents behind interactions with knowledge graph for recommendation, in: *Proceedings of the web conference 2021*, 2021, pp. 878–887, <https://doi.org/10.1145/3442381.3450133>.
- [11] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G.D. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier et al., Knowledge graphs, *ACM Computing Surveys (Csur)* **54**(4) (2021), 1–37, <https://doi.org/10.1145/3447772>.
- [12] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85, <https://doi.org/10.1145/2629489>.
- [13] N. Heist, S. Hertling, D. Ringler and H. Paulheim, Knowledge Graphs on the Web—An Overview, *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges* (2020), 3–22, <https://doi.org/10.3233/SSW200009>.
- [14] M. Färber, A. Rettinger and B. El Asmar, On emerging entity detection, in: *European Knowledge Acquisition Workshop*, Springer, 2016, pp. 223–238, https://doi.org/10.1007/978-3-319-49004-5_15.

Table 8

KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *recall-oriented* mapping scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
Classification	Cities	Accuracy ↑	0.751	0.698	0.760	0.807	0.656	0.760
	Forbes	Accuracy ↑	0.537	0.486	0.590	0.586	0.546	0.556
	AAUP	Accuracy ↑	0.612	0.584	0.588	0.595	0.557	0.634
	MetacriticMovies	Accuracy ↑	0.725	0.709	0.691	0.719	0.710	0.718
	MetacriticAlbums	Accuracy ↑	0.751	0.638	0.625	0.632	0.613	0.657
	ComicCharacters	Accuracy ↑	0.119	0.081	0.478	0.484	0.476	0.475
	MillionSongDataset	Accuracy ↑	0.156	0.139	0.584	0.584	0.588	0.589
Regression	Cities	RMSE ↓	1.318	1.203	0.558	0.539	0.623	0.516
	Forbes	RMSE ↓	1.137	1.156	0.595	0.596	0.607	0.595
	AAUP	RMSE ↓	1.347	1.227	0.572	0.578	0.596	0.540
	MetacriticMovies	RMSE ↓	0.969	0.965	0.482	0.461	0.468	0.462
	MetacriticAlbums	RMSE ↓	0.962	1.035	0.547	0.534	0.546	0.518
Clustering	Teams	Accuracy ↑	0.996	0.954	0.994	0.995	0.995	0.995
		ARI ↑	0.318	0.151	0.061	0.043	0.050	0.090
		NMI ↑	0.260	0.106	0.054	0.040	0.035	0.075
	ComicCharacters	Accuracy ↑	0.115	0.076	0.473	0.460	0.465	0.474
		ARI ↑	0.000	0.000	0.016	0.016	0.014	0.018
		NMI ↑	0.171	0.174	0.008	0.008	0.006	0.006
	CitiesAndCountries	Accuracy ↑	0.933	0.893	0.969	0.791	0.898	0.792
		ARI ↑	0.733	0.676	0.863	0.065	0.609	0.030
		NMI ↑	0.641	0.473	0.756	0.204	0.523	0.199
	Cities2000AndCountries	Accuracy ↑	0.974	0.903	0.934	0.951	0.965	0.936
		ARI ↑	0.900	0.764	0.754	0.814	0.865	0.761
		NMI ↑	0.836	0.573	0.680	0.721	0.783	0.658
	CitiesMoviesAlbums-CompaniesUni	Accuracy ↑	0.880	0.823	0.941	0.963	0.877	0.916
ARI ↑		0.856	0.758	0.871	0.919	0.827	0.871	
NMI ↑		0.743	0.650	0.845	0.882	0.727	0.841	
Document Similarity	LP50	Spearman ↑	0.412	0.342	0.451	0.443	0.444	0.386
		Pearson ↑	0.631	0.596	0.670	0.669	0.659	0.618
		Harm. Mean ↑	0.492	0.431	0.539	0.533	0.531	0.475
Entity Relatedness	KORE	Kendall's Tau ↑	0.386	0.333	0.278	0.221	0.208	0.372
Semantic Analogies	CurrencyEntities	Accuracy ↑	0.355	0.322	0.436	0.038	0.027	0.202
	CityStateEntities	Accuracy ↑	0.440	0.409	0.214	0.444	0.066	0.484
	CapitalCountryEntities	Accuracy ↑	0.990	0.844	0.800	0.925	0.490	0.923
	AllCapitalCountryEntities	Accuracy ↑	0.928	0.818	0.719	0.831	0.624	0.857
Recommendation	MovieLens	F1-score ↑	0.013	0.010	0.011	0.011	0.011	0.016
	LibraryThing	F1-score ↑	0.013	0.013	0.006	0.009	0.007	0.014
	LastFm	F1-score ↑	0.052	0.048	0.021	0.029	0.024	0.041

[15] G. Radevski, K. Gashteovski, C.-C. Hung, C. Lawrence and G. Glavaš, Linking Surface Facts to Large-Scale Knowledge Graphs, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7189–7207, <https://doi.org/10.18653/v1/2023.emnlp-main.445>.

[16] B. Kotnis, K. Gashteovski, D. Rubio, A. Shaker, V. Rodriguez-Tembras, M. Takamoto, M. Niepert and C. Lawrence, MILIE: Modular & Iterative Multilingual Open Information Extraction, in: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 6939–6950, <https://doi.org/10.18653/v1/2022.acl-long.478>.

[17] G. Liu, X. Li, J. Wang, M. Sun and P. Li, Extracting Knowledge from Web Text with Monte Carlo Tree Search, in: *The Web Conference 2020*, 2020, pp. 2585–2591, <https://doi.org/10.1145/3366423.3380010>.

[18] J. Lehmann, R. Isele, M. Jakob et al., DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195, <https://doi.org/10.3233/SW-140134>.

- [19] F. Mahdisoltani, J. Biega and F.M. Suchanek, YAGO3: A knowledge base from multilingual wikipedias, in: *CIDR*, 2013.
- [20] B. Xu, C. Xie, Y. Zhang, Y. Xiao, H. Wang and W. Wang, Learning defining features for categories, in: *25th International Joint Conference on Artificial Intelligence*, 2016, pp. 3924–3930.
- [21] S. Zhang and K. Balog, Web Table Extraction, Retrieval, and Augmentation: A Survey, *ACM Transactions on Intelligent Systems and Technology (TIST)* **11**(2) (2020), 1–35, <https://doi.org/10.1145/3372117>.
- [22] D.B. Lenat, CYC: A large-scale investment in knowledge infrastructure, *Communications of the ACM* **38**(11) (1995), 33–38.
- [23] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.
- [24] H. Paulheim, How much is a triple, in: *International Semantic Web Conference (ISWC)*, 2018.
- [25] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250, <https://doi.org/10.1145/1376616.1376746>.
- [26] A. Pradhan, K.K. Todi, A. Selvarasu and A. Sanyal, Knowledge Graph Generation with Deep Active Learning, in: *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–8, <https://doi.org/10.1109/IJCNN48605.2020.9207515>.
- [27] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L.P. Gilo, D. Dona, O. Corcho and D. Chaves-Fraga, Knowledge graph construction with R2RML and RML: an ETL system-based overview, in: *Second International Workshop on Knowledge Graph Construction*, 2021.
- [28] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana and M.-E. Vidal, SDM-RDFizer: An RML interpreter for the efficient creation of RDF knowledge graphs, in: *Proceedings of the 29th ACM international conference on Information & Knowledge Management*, 2020, pp. 3039–3046, <https://doi.org/10.1145/3340531.3412881>.
- [29] C. Niklaus, M. Cetto, A. Freitas and S. Handschuh, A survey on open information extraction, *arXiv preprint arXiv:1806.05599* (2018).
- [30] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka and T. Mitchell, Toward an architecture for never-ending language learning, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24, 2010, pp. 1306–1313.
- [31] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic web* **8**(3) (2016), 489–508, <https://doi.org/10.3233/SW-160218>.
- [32] M.Y. Jaradeh, K. Singh, M. Stocker, A. Both and S. Auer, Information extraction pipelines for knowledge graphs, *Knowledge and Information Systems* **65**(5) (2023), 1989–2016, <https://doi.org/10.1007/s10115-022-01826-x>.
- [33] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson and J. Taylor, Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it’s done, *Queue* **17**(2) (2019), 48–75, <https://doi.org/10.1145/3331166>.
- [34] F.M. Suchanek, G. Kasneci and G. Weikum, YAGO: a core of semantic knowledge, in: *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706, <https://doi.org/10.1145/1242572.1242667>.
- [35] T. Pellissier Tanon, G. Weikum and F. Suchanek, YAGO 4: A reason-able knowledge base, in: *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, Springer, 2020, pp. 583–596, https://doi.org/10.1007/978-3-030-49461-2_34.
- [36] R.V. Guha, D. Brickley and S. Macbeth, Schema.org: evolution of structured data on the web, *Communications of the ACM* **59**(2) (2016), 44–51, <https://doi.org/10.1145/2844544>.
- [37] J. Hoffart, F.M. Suchanek, K. Berberich and G. Weikum, YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia, *Artificial intelligence* **194** (2013), 28–61, <https://doi.org/10.1016/j.artint.2012.06.001>.
- [38] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel et al., Never-ending learning, *Communications of the ACM* **61**(5) (2018), 103–115, <https://doi.org/10.1145/3191513>.
- [39] R. Navigli and S.P. Ponzetto, BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, *Artificial intelligence* **193** (2012), 217–250, <https://doi.org/10.1016/j.artint.2012.07.001>.
- [40] T. Flati, D. Vannella et al., Two is bigger (and better) than one: the wikipedia bitaxonomy project, in: *52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, 2014, pp. 945–955, <https://doi.org/10.3115/v1/P14-1089>.
- [41] R. Navigli, M. Bevilacqua, S. Conia, D. Montagnini and F. Cecconi, Ten Years of BabelNet: A Survey., in: *IJCAI*, 2021, pp. 4559–4567, <https://doi.org/10.24963/ijcai.2021/620>.
- [42] S. Hertling and H. Paulheim, Dbkwik: extracting and integrating knowledge from thousands of wikis, *Knowledge and Information Systems* **62**(6) (2020), 2169–2190, <https://doi.org/10.1007/s10115-019-01415-5>.
- [43] S. Heindorf, M. Potthast, B. Stein and G. Engels, Vandalism detection in wikidata, in: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 327–336, <https://doi.org/10.1145/2983323.2983740>.
- [44] M. Mintz, S. Bills et al., Distant supervision for relation extraction without labeled data, in: *Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, 2009, pp. 1003–1011.
- [45] N. Heist and H. Paulheim, Transformer-based Subject Entity Detection in Wikipedia Listings, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2022) co-located with the 21th International Semantic Web Conference (ISWC 2022), Virtual Conference, online, October 24, 2022*, M. Alam and M. Cochez, eds, CEUR Workshop Proceedings, Vol. 3342, CEUR-WS.org, 2022. <https://ceur-ws.org/Vol-3342/paper-2.pdf>.
- [46] N. Heist and H. Paulheim, Information extraction from co-occurring similar entities, in: *The Web Conference 2021*, 2021, pp. 3999–4009, <https://doi.org/10.1145/3442381.3449836>.
- [47] N. Heist and H. Paulheim, NASTyLinker: NIL-Aware Scalable Transformer-Based Entity Linker, in: *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, Lecture Notes in Computer Science, Vol. 13870, Springer, 2023, pp. 174–191, https://doi.org/10.1007/978-3-031-33455-9_11.

- [48] N. Heist and H. Paulheim, Uncovering the Semantics of Wikipedia Categories, in: *International Semantic Web Conference*, Springer, 2019, pp. 219–236.
- [49] N. Heist and H. Paulheim, Entity extraction from Wikipedia list pages, in: *European Semantic Web Conference*, Springer, 2020, pp. 327–342, https://doi.org/10.1007/978-3-030-30793-6_13.
- [50] S.P. Ponzetto and R. Navigli, Large-scale taxonomy mapping for restructuring and integrating Wikipedia, in: *21st International Joint Conference on Artificial Intelligence*, 2009.
- [51] M.A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: *14th conference on Computational linguistics-Volume 2*, 1992, pp. 539–545.
- [52] S. Hertling and H. Paulheim, WebIsALOD: providing hypernymy relations extracted from the web as linked open data, in: *International Semantic Web Conference*, Springer, 2017, pp. 111–119, https://doi.org/10.1007/978-3-319-68204-4_11.
- [53] N. Heist, Towards Knowledge Graph Construction from Entity Co-occurrence, in: *Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018)*, Nancy, France, November 13, 2018, CEUR Workshop Proceedings, Vol. 2306, CEUR-WS.org, 2018.
- [54] N. Heist and H. Paulheim, The CaLiGraph Ontology as a Challenge for OWL Reasoners, in: *Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021)*, Virtual Event, October 27th, 2021, CEUR Workshop Proceedings, Vol. 3123, CEUR-WS.org, 2021, pp. 21–31.
- [55] J. Qin and M. Iwaihara, Annotating Column Type Utilizing BERT and Knowledge Graph Over Wikipedia Categories and Lists, in: *DEIM Forum*, 2022.
- [56] R. Biswas, R. Sofronova, M. Alam, N. Heist, H. Paulheim and H. Sack, Do judge an entity by its name! entity typing using language models, in: *The Semantic Web: ESWC 2021 Satellite Events: Virtual Event, June 6–10, 2021, Revised Selected Papers 18*, Springer, 2021, pp. 65–70, https://doi.org/10.1007/978-3-030-80418-3_12.
- [57] S. Chowdhury, M. Ebrahimi, A. Eberhart and P. Hitzler, Memory Networks for RDFS reasoning: Experiments, in: *Joint Proceedings of SemREC 2022 and SMART 2022 co-located with 21st International Semantic Web Conference (ISWC 2022)*, Hybrid event, Hangzhou, China, October 24-27, 2022, CEUR Workshop Proceedings, Vol. 3337, CEUR-WS.org, 2022, pp. 28–32.
- [58] J. Andersen, S. Cazalens and P. Lamare, Assessing Knowledge Graphs Accountability, in: *2023 Extended Semantic Web Conference (ESWC23)*, 2023, https://doi.org/10.1007/978-3-031-43458-7_7.
- [59] K. Janowicz, P. Hitzler, B. Adams, D. Kolas and C. Vardeman II, Five stars of linked data vocabulary use, *Semantic Web* 5(3) (2014), 173–176, <https://doi.org/10.3233/SW-140135>.
- [60] M. Färber, F. Bartscherer, C. Menne and A. Rettinger, Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago, *Semantic web* 9(1) (2017), 77–129, <https://doi.org/10.3233/SW-170275>.
- [61] N. Heist, S. Hertling and H. Paulheim, KGrEaT: A Framework to Evaluate Knowledge Graphs via Downstream Tasks, in: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 3938–3942, <https://doi.org/10.1145/3583780.3615241>.
- [62] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, *Advances in neural information processing systems* 26 (2013).
- [63] B. Yang, W.-t. Yih, X. He, J. Gao and L. Deng, Embedding entities and relations for learning and inference in knowledge bases, *arXiv preprint arXiv:1412.6575* (2014).
- [64] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier and G. Bouchard, Complex embeddings for simple link prediction, in: *International conference on machine learning*, PMLR, 2016, pp. 2071–2080.
- [65] P. Ristoski and H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I* 15, Springer, 2016, pp. 498–514, https://doi.org/10.1007/978-3-319-46523-4_30.
- [66] S. Hertling and H. Paulheim, Transformer Based Semantic Relation Typing for Knowledge Graph Integration, in: *European Semantic Web Conference*, Springer, 2023, pp. 105–121, https://doi.org/10.1007/978-3-031-33455-9_7.
- [67] F. Ilievski, P. Szekely and B. Zhang, Cskg: The commonsense knowledge graph, in: *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, Springer, 2021, pp. 680–696, https://doi.org/10.1007/978-3-030-77385-4_41.
- [68] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmman, S. Sun and W. Zhang, Knowledge vault: A web-scale approach to probabilistic knowledge fusion, in: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 601–610, <https://doi.org/10.1145/2623330.2623623>.