Construction of Knowledge Graphs: State and Challenges

Marvin Hofer [b,*], Daniel Obraczka [b], Alieh Saeedi [a,b], Hanna Köpcke [a] and Erhard Rahm [a,b]

[a] *Dept. of Computer Science, Leipzig University, Germany*

[b] *Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany*

Abstract. With knowledge graphs (KGs) at the center of numerous applications such as recommender systems and question answering, the need for generalized pipelines to construct and continuously update such KGs is increasing. While the individual steps that are necessary to create KGs from unstructured (e.g. text) and structured data sources (e.g. databases) are mostly well-researched for their one-shot execution, their adoption for incremental KG updates and the interplay of the individual steps have hardly been investigated in a systematic manner so far. In this work we first discuss the main graph models for KGs and introduce the major requirement for future KG construction pipelines. We then provide an overview of the necessary steps to build high-quality KGs including cross-cutting topics such as metadata management, ontology development and quality assurance. We then evaluate the state of the art of KG construction w.r.t the introduced requirements for specific popular KGs as well as some recent tools and strategies for KG construction. Finally we identify areas in need of further research and improvement.

Keywords: Knowledge Graph, Data Integration, Data Science

1. Introduction

Aggregated machine-readable information in the form of knowledge graphs (KG) serves as the backbone of numerous data science applications nowadays, ranging from question answering [1] over recommendation systems [2] to predicting drug-target interactions [3]. The ever-changing nature of information necessitates the design of KG construction pipelines that are able to incorporate new information continuously. In the development of such a system, knowledge engineering teams have to deal with a variety of challenges from tackling scalability and heterogeneous data sources to tracking the provenance of data. Given the usually large volume of data that needs to be integrated, such pipelines have to be automatized as much as possible while aiming at a high degree of data quality.

Knowledge graphs generally integrate heterogeneous data from a variety of sources with unstructured and semi-structured data of different modalities (e.g. pictures, audio, text) as well as structured data such as databases or other KGs in a semantically rich way. The construction of a KG therefore encompasses a multi-disciplinary effort requiring expertise from research areas such as natural language processing (NLP), data integration, knowledge representation and knowledge management. Knowledge graphs are at the center of numerous use-cases for data analysis and decision support. In the clinical setting, enriching patient data with medical background knowledge enables improved clinical decision support [4]. Sonntag et. al. [5] argue that properly aligning the semantic labels attached to the patient data with medical ontologies is crucial in creating meaningful access to the heterogeneous patient data. Knowledge graphs are also used to organize the relevant information for fast emerging global topics, such as pandemics (e.g., Covid-19) or natural disasters. Machine learning also benefits from KGs as a source of

*Corresponding author. E-mail: hofer@informatik.uni-leipzig.de.

labeled training data or other input data [6, 7] thereby supporting the development of knowledge- and data-driven AI approaches [8].

Improving the pipelines that build these knowledge graphs and enabling them to efficiently keep current and semantically meaningful aggregated knowledge is therefore an effort that benefits a wide range of application areas. However, KGs are currently created often in a batch-like manner so that the respective pipelines are unfit to incorporate new incoming facts into a KG without full re-computation of the individual tasks. Furthermore, different steps of the pipelines often require manual intervention thereby limiting scalability to large data volumes and increasing the time for updating a KG.

There is a growing number of surveys about knowledge graphs especially on their general characteristics and usage forms [6–8]. An excellent tutorial-style overview about the construction and curation of KGs is provided in [9] with a focus on integrating data from textual and semi-structured data sources such as Wikipedia. Other surveys focus on KG construction with specific technologies [10, 11] or only a single domain such as for geographical data [12]. The recent work of [13] analyzes existing KG construction approaches and propose a general construction pipeline that we will discuss in Section 3. In this paper we build on these previous studies for KG construction but aim at a broader analysis by introducing main requirements for KG construction and covering different graph data models, incremental KG construction and data integration including incremental entity resolution in much more detail. Specifically, our contributions are the following:

- After a definition of KGs and a comparative discussion of graph data models for KGs we introduce and categorize the general requirements for incremental KG construction in Section 2.
- In Section 3 we provide an overview over the main tasks in incremental KG construction pipelines and proposed solution approaches for them.
- We then investigate and compare existing construction efforts for selected KGs as well as within recent tools for KG construction w.r.t. the requirements introduced earlier. This also allows us to identify tasks that are not yet supported well.
- Section 5 discusses open challenges for KG construction.

Finally, we provide closing thoughts and a summary in the last section.

2. KG background and requirements for KG construction

We first outline the notion of knowledge graph (KG) used in this paper which is based on the integration of information from multiple sources. We then briefly introduce and compare the two most popular graph data models for KGs, namely RDF and property graphs. Finally we outline the main requirements or desiderata for largely automatic construction and maintenance of KGs.

2.1. Knowledge Graph

KGs realize a physical data integration where the information from different sources is combined in a new logically centralized graph-like representation. KGs are schema-flexible and the graph structure allows a relatively easy addition of new entities and their interlinking with other entities. This is in contrast to the use of data warehouses as a popular approach for physical data integration. Data warehouses focus on the integration of data within a structured (relational) database with a relatively static schema that is optimized for certain multi-dimensional data analysis. Schema evolution is a manual and tedious process making it difficult to add new data sources or new kinds of information not conforming to the schema.

Although the term *knowledge graph* goes back as far as 1973 [14], it gained popularity through the 2012 blog post[1] about the Google KG. Afterwards, several related definitions of knowledge graphs were proposed, either in research papers [7, 9, 15–17] or by companies using or supporting KGs (OpenLink, Ontotext, Neo4J, TopQuadrant,

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

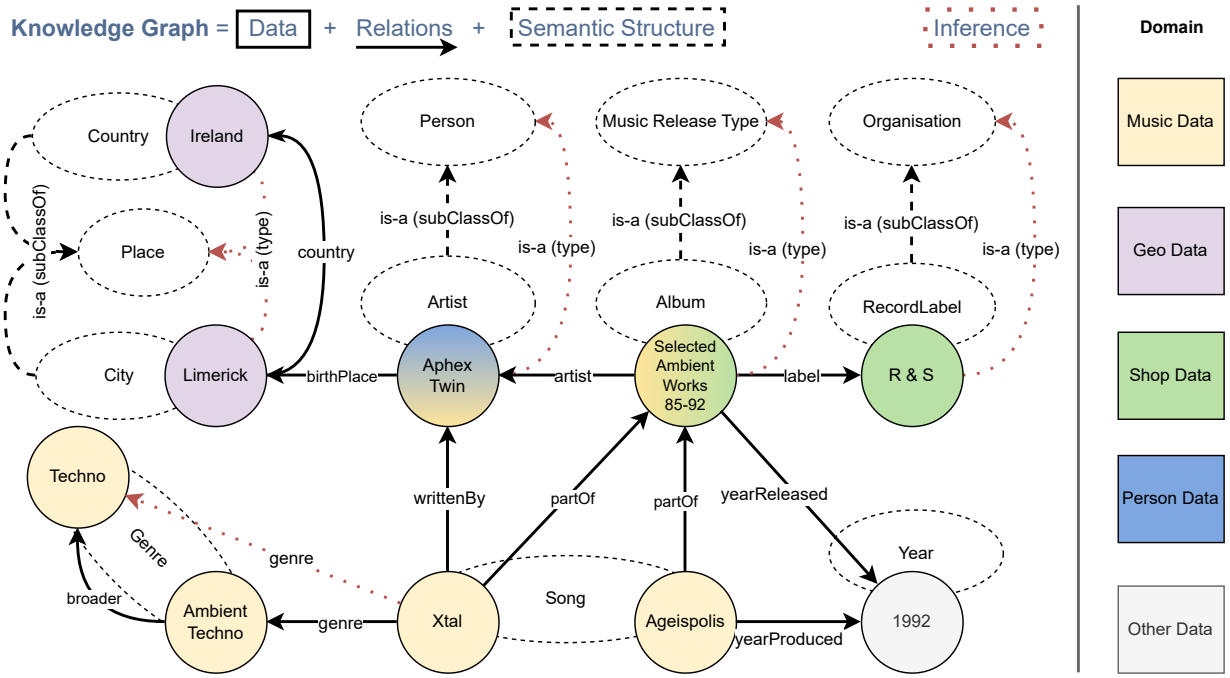

Fig. 1. **Simplified KG example** with integrated data from several domains. Entities and relations are described by an underlying ontology that allows the inference of additional relations (dashed red lines).

Amazon, Diffbot[2], Google). Ehrlinger et al. [16] give a comprehensive overview of KG definitions and provide their own: "A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge." Hogan et. al. [18] argue that this definition is very specific and excludes various industrial KGs which helped to popularize the concept.

We therefore define KGs more inclusively as a graph of data consisting of semantically described entities and relations of different types that are integrated from different sources. Entities have a unique identifier. KG entities and relations are semantically described using an ontology or, more clearly, an ontological representation [19]. An ontological theory can define formal naming, category systems, properties, and relations between concepts, data, and entities that encompass one or several domains of discourse. For this purpose common ontology relations such as *is-a (subclass-of)* and *has-a* relationships should be supported. Furthermore, the ontology should enable to infer implicit knowledge from the explicitly represented information [17].

Figure 1 visualizes a simplified KG example with integrated information from several domains where ontological information such as types or is-a relations are dashed. There are ten entities of the following eight types: Country (*Ireland*), City (*Limerick*), Artist (*Aphex Twin*), Album (*Selected Ambient Works 85-9*), Record Label (*R & S*), Genre (*Techno*, *Ambient Techno*), Song (*Xtal*, *Ageispolis*) and Year (*1992*). Ontological *is-a* (sub-class) relations interrelate City and Country with Place, Artist with Person, Album with Music Release Type, and Record Label with Organisation. The domain is further described by the named relationships: country, birthPlace, artist, label, writtenBy, yearReleased, founded, broader, genre, yearProduced, partOf. Based on the given relationships and typing, further information is inferable, e.g., Aphex Twin's broader birth place is Ireland, the song Xtal is also of genre Techno, Aphex Twin being of the type Artist means this instance is also of the type Person (for readability not all possible inferences are denoted).

Table 1
Comparison of RDF and property graphs.

	Resource Description Framework (RDF)	Property Graph Model (PGM)
base constructs	triples <subject, predicate, object>	labeled vertices and edges and their properties
entity identity	URI	local (implementation specific)
node classification	rdf:type triples	type labels
ontology support	RDFS, OWL2 vocabularies	limited, e.g., schema graph
integrity constraints	SHACL, SHEX	PG-Keys
query language	SPARQL(-Star)	Cypher, Gremlin, G-Core, PGQL
exchange format	N-Triples, N-Quads, (RDF/XML, JSONLD)	application specific e.g., PGEF, GDL
meta information	reification, singleton-property, (RDF-Star)	dedicated properties

2.2. Graph Models

To represent and use KGs as informally defined above, a powerful graph data model is needed that supports entities and relations of different types as well as their ontological description and organization [20]. Moreover, the graph data model should provide a comprehensive query language and possibly more advanced graph analysis or mining capabilities, e.g., for clustering of similar entities or determining graph embeddings for machine learning tasks. Support for integrity constraints is also desirable to automatically control the consistency and therefore quality of graph data to some extent. Furthermore, it should be possible to represent annotating metadata of KG entities, e.g., about their origin and transformation during KG construction. Additionally, it is desirable to reflect the development of the KG over time so that a temporal KG analysis is supported. This can be achieved by a temporal graph data model with time metadata for every entity and relation and temporal query possibilities, e.g., to determine previous states of the KG or to find out what has been changed in a certain time interval. The temporal development of a KG might alternatively be reflected with a versioning concept where new KG versions are periodically released. Finally, the graph data model should facilitate the KG construction process and its different tasks for acquiring, transforming and integrating heterogeneous data from different sources. This can be supported by suitable formats to seamlessly exchange data of the chosen graph model between different steps and processing nodes of a KG construction pipeline.

The most common graph models used for KGs are the Resource Description Framework and the Property Graph Model. In the following, we briefly describe both and discuss how they meet the introduced desiderata. Table 1 summarizes some of the key differences of both models. At the end we also contrast the different terminology of the models and specify the terms used in the rest of this paper.

Resource Description Framework (RDF) [21]

RDF is the data exchange format of the Semantic Web, and was initially developed to describe metadata of web-resources. Today, the W3C proposes many technologies around RDF that help build and use knowledge graphs either as part of the Linked Data Cloud or in an encapsulated environment. KGs are represented by a set of <subject, predicate, object> triples that uniformly represent named relations (predicates) of entities (subjects) to either attribute values (literals) or other entities. Entities are assigned a URI-based identifier that can refer to either a global or local namespace. Sets of triples may also be grouped within named graphs to aggregate more information by extending the triple structure to quads of the form <subject, predicate, object, named-graph>. Standard RDF does not support edge properties although the RDF-Star extension[3] includes a similar feature, where single triples are usable in the head or tail part of another triple.

The RDF standard also defines vocabularies such as RDF Schema (RDFS) to further express semantic structure by allowing the definition of classes, properties and their hierarchies. An RDF resource's type (or entity class) is assigned by using the standard RDF vocabulary[4] to define triples of the form `<s rdf:type o>` where o is the class (type) of the resource. In addition to RDFS, a widely used approach for defining ontologies is the Web

[2]https://blog.diffbot.com/knowledge-graph-glossary/
[3]https://www.w3.org/2022/08/rdf-star-wg-charter/
[4]http://www.w3.org/1999/02/22-rdf-syntax-ns#

Ontology Language (OWL), more specifically the current version OWL 2, which adds semantics to the data, using a variety of axioms.

Besides syntax validation (triples/quads, URIs, datatypes) RDF triple stores do not provide a standard method to define and validate graph data integrity or shape constraints (similar to relation database schemata). Therefore overlaid solutions such as SHACL (Shape Constraint Language [22]) or ShEx (Shape Expressions [23]) are developed that can be used to validate the semantic correctness of the graphs structure, node or property constraints, cardinalities, and other constructs.

While some RDF stores or triple stores are built from scratch to optimize the management of RDF triples, others might use existing SQL or NoSQL systems in the underlying database processing layer. The primary query language for RDF (moreover, the Semantic Web) is the standardized language SPARQL[5], with an extended version for RDF-Star called SPARQL-Star. Standard exchange formats for RDF are N-Triples, N-Quads, Turtle, or adapted syntax formats like RDF/XML and JSON-LD.

There are different possibilities to assign metadata to entities, relations and properties like using RDF-Star or named graphs as explained and evaluated in [24, 25]. Another method is to use a combination of entity or property identifiers and referencing its hash-sum as key in another database to attach metadata information. The usage of such support constructs for metadata management generally increases complexity of the graph structure and queries and can possibly increase processing time.

There is some work around the representation, querying storage and other aspects of temporal information in RDF [26]. The investigated methods focus on different temporal granularity and dimensions, including approaches that target to query single snapshots, time windows or to inspect the evolution of temporal graphs.

As many knowledge graphs are in RDF, several frameworks have been developed to perform graph analytics, algorithms, or mining tasks using RDF as input [27].

Property Graph Model (PGM) [28]

The property graph data model, also called Labeled Property Graph (LPG), supports the flexible definition of graph structures with heterogeneous nodes (vertices) and directed edges to represent entities of different kinds, and the relationships between them. Both nodes and edges can have multiple (type-)labels expressing their role in the modeled domain, e.g., *User* as a node label and *follows* as edge label. Additionally, properties (in the form of key-value pairs) can be assigned to both nodes and edges. Further, in the most common implementations, vertices and edges are specified by a unique identifier. While label and property names represent some schema-like information, there is intentionally no predefined schema to allow flexible incorporation of heterogeneous entities and relations of different kinds (although a schema graph can be inferred from the type information [29]). There is no built-in support for ontologies, e.g., to provide is-a relations between entity categories. Embedded metadata can be relatively easy maintained for entities and relationships by using dedicated properties, e.g. for provenance or time annotations.

In contrast to RDF, the PGM with its vertices, edges and properties is more related to graph models in graph theory thereby contributing to their good understandability. As there is not yet a global (defacto) standard for PGM, its capabilities highly depend on its implementation. The PGM is increasingly popular in research and practice and supported by several graph database systems, such as Neo4j [30], JanusGraph [31] or TigerGraph [32], and processing frameworks, such as Oracle Labs PGX [33] or Gradoop [34]. It is further the base data model for several graph query languages [35], such as G-Core [36], Gremlin [37], PGQL [38], Cypher [39], as well as SQL/PGQ and GQL [40], the upcoming ISO standard language for property graph querying. Efforts on a standardized PGM serialization format, are the JSON-based Property Graph Exchange Format (PGEF) [41], YARS-PG [42] or the Graph Definition Language (GDL)[6].

Similar to RDF stores, data integrity of PGM databases is generally limited to syntax or basic value constraints. A first effort about the aspects of property graph *key constraints* is proposed by Angles et al. [43] by identifying four natural key types: identifier, exclusive mandatory, exclusive singleton, or exclusive.

There are several extensions to the PGM for supporting temporally evolving graph data [34, 44] and graph streams [45], often with advanced analysis capabilities for graph mining.

[5]https://www.w3.org/TR/sparql11-overview/

[6]https://github.com/dbs-leipzig/gdl

Discussion As outlined above there are pros and cons for both RDF and PGM. The intensive use of RDF in the Semantic Web and Linked Open Data communities has lead to its wide-spread application for KGs; in fact most KGs we will consider in Section 4 are using RDF. The triple-based graph representation of RDF is quite flexible and allows a uniform representation for entities and relationships. But it also is hard to understand for users as the information of an entity is distributed over many triples. Although, RDF-Star greatly improves the formal meta expressiveness of RDF, specific cases are still not presentable as in PGM, without utilizing support constructs. In the PGM we can have two equally named but independently addressable relations between two entities, both with individually resolvable edge properties and the issue of interference. However, in RDF-Star, triples (relations) always identify based on their comprised elements, and it is not possible to attach distinguishable sets of additional data to equally named relations without overlapping or utilizing support constructs [46]. The PGM has become increasingly popular for advanced database and network applications (e.g., to analyse social networks) but its limited ontology support has so far hindered its broader adoption for KGs.

Besides, RDF (direct graphs) or property graphs, in some cases custom models or special high arity representation could be used to cover specific features, like access-levels, temporal information, or multihop relations in one record (node-edge-edge-node) [47]. However, the usage of such custom models will lower interoperability with existing tools (requiring transformation), and complicate its own reusability by others.

The decision between the use of RDF and PGM (or a custom data model) depends on the targeted application or use case of the final knowledge graph. Lassila et al [46] conclude that both formats are qualified to meet their challenges and none of the two is perfect for every use case. They thus recommend to increase interoperability between both models to reuse existing techniques of both approaches. Various efforts to address this problem have been made in the recent years. The Amazon Neptune[7] database service allows users to operate PGM and RDF interchangeably. Hartig et al. [48] and Abuoda et al. [49] discuss transformation strategies between RDF and PGM to lower usage boundaries.

Terminology

Due to the different communities around PGM and RDF there are many similar but differently named terms in use. Table 2 lists some of the terms that we will use as synonymous in this paper with the underlined ones used preferably.

Table 2
Synonymously used KG terms in RDF and PGM.

Terms	Description
entity, instance, subject & object & resource (RDF), individual	KG nodes that represent a specific real-world or abstract thing
relation, property (RDF)	a relationship (edge, link) between two KG entities.
type, class, label, concept	Identifier that represents the same kind or group of entities or relations.
property (PGM), attribute (RDF)	an entity feature identifier pointing to a value
property value, literal, attribute value	any value that is not referable to as an entity.

Furthermore, we refer to the smallest unit of information as *statement* or *fact*. For RDF this would describe a triple, for PGM this can be assigning a property(-value), adding a type label to an entity or adding a relation between two nodes.

2.3. Requirements of KG construction

The development and maintenance of KGs encompasses several steps to integrate relevant input data from different sources. While the specific steps depend on the input data to be integrated and the intended usage forms of the KG, it is generally desirable that the steps are executed within pipelines with only a minimum of manual interaction and curation. However, a completely automatic KG construction is not yet in reach since several steps (e.g., identification of relevant sources, development of the KG ontology), as we will see, typically require human input, either by individuals, expert groups or entire communities [50].

[7] https://aws.amazon.com/en/blogs/aws/amazon-neptune-a-fully-managed-graph-database-service/

The KG construction process should result in a high-quality KG based on an expressive *KG data model* as discussed above. The quality of a KG (and data sources) can be measured along several dimensions such as correctness, freshness, comprehensiveness and succinctness [51, 52]. The correctness (accuracy, consistency) aspect implies that each entity, concept, relation and property is *canonicalized* by having a unique identifier and being included exactly once [9]. The freshness aspect requires a continuous update of the instance and ontological information in a KG to incorporate all relevant data source changes. The comprehensiveness requirement asks for a good coverage of all relevant data and that complementing data from different sources is combined [52]. Finally, the succinctness criterion asks for a high focus of the data (e.g., on a single domain) [7] and exclusion of unnecessary data which also improves resource consumption and scalability of the system.

In the following we discuss general requirements or desiderata for KG construction and maintenance in more detail as they should guide the realization of suitable implementation approaches. We group these requirements into four aspects related to 1) input consumption, 2) incremental data processing capabilities, 3) tooling/pipelining, and 4) quality assurance, whereas some essential prerequisites can affect multiple parts of the workflow (e.g., supportive metadata). Please note, that we outline the desired functionality for defining arbitrary KG pipelines and that only a subset of it is typically needed for a specific KG project.

- **Input Data requirements**. It should be possible to integrate a large number of data sources as well as a high amount of data (data scalability). There should also be support for heterogeneous and potentially low-quality input data of different kinds such as structured, semi-structured and multimodal unstructured data (textual documents, web data, images, videos, etc.). As a result, KG construction requires scalable methods for the acquisition, transformation and integration of these diverse kinds of input data. The processing of semi-structured and unstructured data introduces the need for knowledge extraction methods to determine structured entities and their relations as well as their transformation into the KG graph data model. Data integration and canonicalization involves methods to determine corresponding or matching entities (entity linking, entity resolution) and their combination into a single representation (entity fusion) as well as matching and merging ontology concepts and properties. For incremental KG construction, the input is not limited to the new data to be added but also includes the current version of the KG and reusable data artifacts such as previously determined mappings specifying how to transform input data into the format of the KG graph model.
- **Support for incremental KG updates**. It should be possible to process the input data both in a batch-like mode where all (new) input data is processed at the same time or in a streaming manner where new data items can continuously be ingested. The initial version of the KG is typically created in a batch-like manner, e.g., by transforming a single data source or by integrating several data sources into an initial KG. After the initial KG version has been established, it is necessary that the KG can be updated to incorporate additional sources and information. A simple approach would perform these updates by a complete recomputation of the KG with the changed input data similar than for the creation of the initial KG. However, such an approach would result in an enormous amount of redundant computation to repeatedly extract and transform the same (unchanged) data and to perform data integration and removal of inconsistencies again, possibly with repeated manual interactions. These problems increase with the number and size of input sources and thus limit or prevent scalability. Hence we require support for incremental KG updates that can either periodically be performed in a batch-like manner or in a more dynamic, streaming-like fashion. The batch approach would not require to completely rebuilt the KG, but focus on adding the new information without reprocessing previously integrated data. A given KG can also be continuously updated with new data in a streaming manner to always provide the most current information for high data freshness. Batch and stream-oriented updates may also be applied in combination [47]. As a result, several pipelines may be needed for the creation of the initial KG, the integration of sources with heterogeneous structure, and different forms of incremental KG maintenance.
- **Pipeline and Tool Requirements**. It should be easy to define and run powerful, efficient and scalable pipelines for creating and incrementally updating a KG. This requires a set of suitable methods or tools for the different steps (discussed in the next section) that should have good interoperability, a good degree of automation, but still support high customizability, and adapt to new domain requirements. While the usage of a uniform KG data model (or serialization) can lower debugging complexity of the workflow, reusing existing toolsets might require transformation/mapping between data formats and the processing steps. Moreover, a pipeline tool should

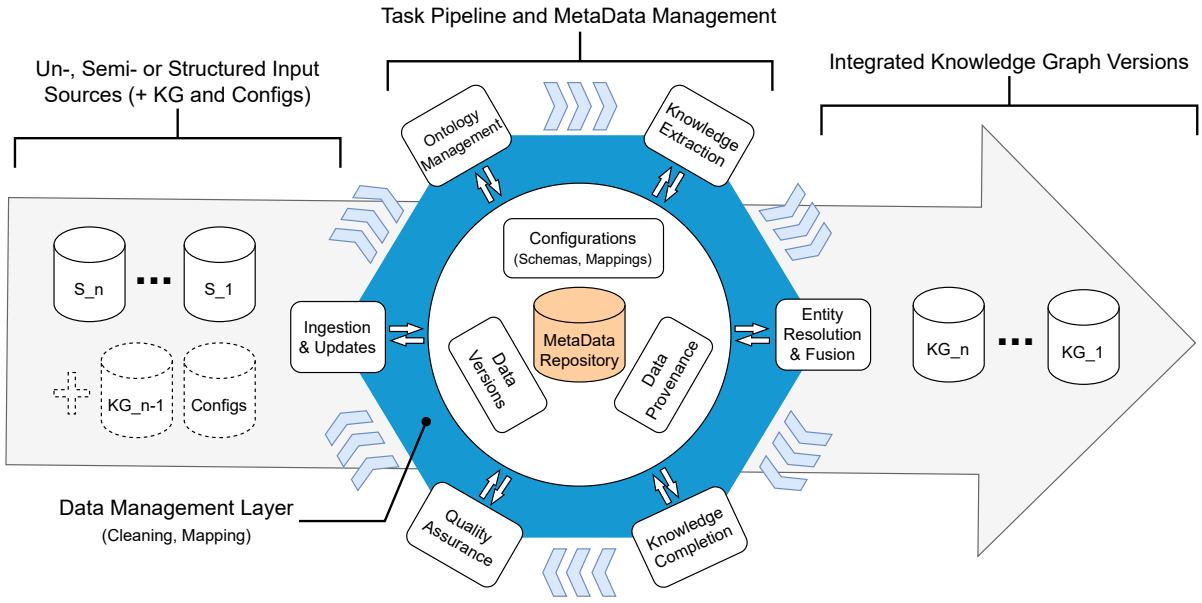

Fig. 2. Incremental Knowledge Graph Construction Pipeline

be provided that can integrate the different tools and manages intermediate results and common metadata, e.g., about provenance. The pipeline tool should further provide administration functionality to design and execute pipelines, to support error handling, performance monitoring and tuning etc. Pipeline processes should scale horizontally as new input data is ingested and the KG size increases over time. Modular processing workflows with transparent interfaces can increase reusability of alternative tools (implementations).

– **Quality Assurance**. Quality assurance is a cross-cutting topic playing an important role throughout the whole KG construction process. Quality problems in the KG can be multi-faceted relating to the ontological consistency, the data quality of entities and relations (comprehensiveness), or domain coverage. The coverage aspect may focus on the inclusion of relevant data and the exclusion of unnecessary data. In some scenarios, the timeliness of data can play a critical role in real-time oriented use cases. If not handled, quality problems might aggravate over time due to the continuous integration of additional data. Therefore methods are needed to evaluate the quality of each step of the construction pipeline as well as of the resulting KG. A specific quality aspect is to validate the KG's data integrity concerning its underlying semantic structure (ontology). Another relevant criteria could be to optimize data freshness to guarantee up-to-date results in upstream applications. Debugging capabilities based on sufficient metadata are helpful to locate the exact points in the construction pipeline where quality problems arise. Methods are then required for fixing or mitigating the detected quality issues by refining and repairing the KG.

3. Construction Tasks

We give an overview over the main tasks for Knowledge Graph Construction with a focus on (semi-)automatic and incremental solutions. In particular we cover the following tasks that often involve several subtasks:

– Data Acquisition & Preprocessing: Selection of relevant sources, acquisition and transformation of relevant source data, initial data cleaning.
– Metadata Management: Acquisition and management of different kinds of metadata, e.g., about the provenance of entities, structural metadata, temporal information, quality reports or process logs.
– Ontology Management: Creation and incremental evolution of a KG ontology.

- Knowledge Extraction (KE): Derivation of structured information and knowledge from unstructured or semi-structured data. using techniques for named entity recognition, entity linking and relation extraction. If necessary this also entails canonicalization of entity and relation identifiers.
- Entity Resolution (ER) and Fusion: Identification of matching entities and their fusion within the KG.
- Quality Assurance (QA): Possible quality aspects, their identification, and repair strategies of data quality problems in the KG.
- Knowledge Completion: Extending a given KG, e.g., by learning missing type information, predicting new relations, and enhancing domain-specific data (polishing).

Figure 2 illustrates a generic pipeline to incrementally incorporate updates from several sources into a KG that may result in a sequence of distinct KG versions. It is important to note, that a construction pipeline does not necessarily follow a fixed execution order for the individual tasks and that not all steps may be required depending on the KG use case. This is also because the required tasks depend on the type of source input. Knowledge extraction is commonly applied on unstructured data inputs like text and may not be needed for structured data, e.g. from databases or other knowledge graphs. Furthermore, the entity linking part of knowledge extraction can make an additional entity resolution step unnecessary. As a result there may be different KG construction pipelines for different use cases and data sources. The steps of Quality Assurance and KG completion to improve the current version of the KG are not needed for every KG update but may be executed asynchronously, e.g., within separate pipelines (although QA actions such as data cleaning also apply to individual tasks). Furthermore, data and metadata management play a special role compared to the other tasks, since they are necessary throughout the entire pipeline therefore representing a cross-cutting task, as indicated by the central position of metadata management in Figure 2.

Tamašauskaitė et. al. [13] propose a similar KG development life-cycle consisting of six main steps with several possible sub tasks. Our work covers their construction tasks and possible solutions in more detail and complements them with other relevant main tasks such as metadata management and the discussion of temporal aspects and versioning. Furthermore, we focus much more on the necessity of enabling incremental updates in the KG construction process. This is because our work is driven by the identified requirements for KG construction, while Tamašauskaitė et. al. try to summarize existing KG construction approaches which often do not support or emphasize incremental updates or are limited to single construction steps.

3.1. Data Acquisition & Preprocessing

3.1.1. Source Selection & Filtering

In order to integrate data into a KG, relevant sources must first be identified. Furthermore, relevant subsets of a data source have to be determined as it is generally unnecessary to integrate all information of a source for a given KG project. For example, for a pandemic-specific KG only health-related parts of existing KGs such as DBpedia may be needed. If the system is not supposed to integrate data sources in their entirety, it can determine a relevant subset by the quality or trustworthiness of a source [53] as well as the importance of single entities [47]. Formulating the quantification [54] of all these criteria alongside computing the cost of integrating a source leads to saving a considerable amount of unnecessary effort while producing a high-quality KG.

Selecting relevant data sources and their subsets are typically manual steps but can be supported by data catalogs providing describing metadata about sources and their contents. Common approaches to determine such metadata are to employ techniques for data profiling, topic modelling, keyword tagging, and categorization [55, 56]. In [9] it is recommended to start KG construction with large curated ("premium") data sources such as Wikipedia and other KGs such as DBpedia. Then further data sources should be identified and integrated to cover additional entities and their relations, especially rather special entities in the "long tail" (e.g., less prominent persons). Given that sources can differ enormously in size and quality, the order in which sources (and their updates) are integrated can have a strong influence on the final quality [57–59]. Especially for creating the initial KG these choices are often crucial. To limit these effects it is advisable to first integrate the sources of the highest quality [57] such as the mentioned premium sources. Nevertheless, ideally integration order should not matter in a high quality pipeline.

3.1.2. Data Acquisition

The data sources of a KG may come in many different data formats such as CSV, XML, JSON, or RDF to meet the requirement of different originating environments and applications. Furthermore, there are different technologies to exchange or acquire data artifacts by providing downloadable files, deploying databases, or supporting access interfaces (APIs). Hence, KG construction has to deal with these heterogeneous data formats and access technologies to acquire the data to be integrated. A common access approach is the use of an adapter component for each source dataset. Such an adapter approach is typical for data integration and there are also supporting tools for use in KG management [60–62].

In addition, KG construction has to deal with continuously changing sources, which necessitates the recognition of such changes and possibly maintaining snapshots of already acquired versions of source data. Possible solutions for change detection include manual user notifications over email, accessing a change API using publish-subscribe protocols [63], or computing diffs by repeatedly crawling external data and comparing it with a previously obtained snapshot.

For RDF stores several strategies for maintaining versions of extracted data have been proposed. With full materialization, complete versions (snapshots) of source data are maintained [64]. With the delta-based strategy only one full version of the dataset needs to be stored and for each new version only the set of changes or deltas has to be kept [65–68]. The annotated triples strategy is based on the idea of augmenting each triple with its temporal validity [69]. Hybrid strategies have also been considered [70].

3.1.3. Transformation & Mapping

A KG construction pipeline has to transform the input data into the final KG data format such as RDF or a property graph format. Furthermore the different pipeline steps may consume and produce different formats so that additional data format transformations or conversions may become necessary. For example, knowledge extraction methods typically process document data such as HTML or Unicode-encoded text while an entity resolution task may require input data in CSV or JSON format.

Data format transformations have especially been addressed for structured data and there exist many tools for this purpose. Depending on the required input format the transformation can be done automatically using generic approaches or requires the manual specification of mappings. Mapping languages allow the specification of complex and reusable mappings, for example to transform relational databases (RDB) into an equivalent RDF representation, e.g. using the R2RML language [71]. RML [72] (RDF mappings language) extends R2RML and allows defining mappings not only from RDB but also from other semi-structured data like XML, TSV/CSV, and JSON. Relatively little work has so far investigated the transformation of structured data into property graphs [73, 74] although the conversion between RDF and property graphs has received some attention [48, 49, 75]. In case of an existing KG as input, a standard solution for RDF to RDF mappings is to use SPARQL-CONSTRUCT[8] queries, which return a single RDF graph by substituting variables of a given graph pattern with the results of the SPARQL query. GQL will support a similar feature for the PGM [40].

3.1.4. Data Cleaning

Data cleaning deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. Whenever possible, data quality problems within the input sources should be handled already during the import process to avoid that wrong or low-quality data is added to the KG. Data cleaning has received a large amount of interest, especially for structured data, in both industry and research and there are numerous surveys and books about the topic, e.g. [76–79]. There are many different kinds of data errors and quality problems to handle such as missing or wrong data values (e.g., due to typos), inconsistent value pairs (e.g. zip code and city), mixing several attribute values in a single freetext attribute (e.g. address or product information), duplicate or redundant information etc. Typically, data cleaning involves several subtasks to deal with these problems, in particular data profiling to identify quality problems [80], data repair to correct identified problems, data transformation to unify data representations and data deduplication to eliminate duplicate entities. Outlier detection is an important kind of data profiling to identify errors in the data based on the assumption of specific "normal" data values. For example,

[8]https://www.w3.org/TR/rdf-sparql-query/#construct

it is unlikely that a person whose birthday was in the middle of the 19th century has died in the year 2020. In order to find logical errors one can use rules or integrity constraints; we discuss such techniques for assuring the quality of a KG in Section 3.6. We outline approaches to identify duplicates across data sources (entity matching) in Section 3.5. Deduplication within a source involves similar steps and it is beneficial to remove duplicates within a source early on to simplify the deduplication across sources. The use of machine learning for data cleaning has become an important trend in recent years as it promises to simplify the complex configuration of the different subtasks. For example HoloClean [81] uses observed data to build a probabilistic model for predicting unknown data values. Other uses of machine learning for data cleaning are covered in [78, 82].

3.2. Metadata Management

Metadata describes data artifacts and is important for the findability, accessibility, interoperability and (re-)usability of these artifacts [12, 83, 84]. There are many kinds of metadata in KGs such as descriptive metadata (content information for discovery), structural metadata (e.g. schemas and ontologies), and administrative metadata concerning technical and process aspects (e.g., provenance information, mapping specifications) [85–87]. It is thus important that KG construction supports the comprehensive representation, management and usability of the different kinds of metadata. From the perspective of KG construction pipelines, this includes metadata for each data source (schema, access specifications), each processing step in the pipeline (inputs including configuration, outputs including log files and reports), about intermediate results and of course the KG and its versions. Moreover, for each fact (entity, relation, property) in the KG there can be metadata such as about provenance, i.e., information about the origin of data artifacts. Such fact-level provenance is sometimes called *deep* or *statement-level provenance*. Examples of deep provenance include information about the creation date, confidence score (of the extraction method) or the original text paragraph the fact was derived from. Such provenance can help to make fact-level changes in the KG without re-computing each step or to identify how and from where wrong values were introduced into the KG [84].

Metadata can be created either manually by human users (e.g., to specify a license for data usage or a configuration of a pipeline step) or by a computer program based on a heuristic or an algorithm [86]. In the latter case the results may be exact or only approximate. For example, data profiling computes accurate statistical information (e.g., about the distribution of values) while the use of machine learning (e.g., for type recognition) usually does not provide perfect accuracy.

To make best use of metadata for KG construction asks for the use of a metadata repository (MDR) to store and organize the different kinds of metadata in a uniform and consistent way [83]. The MDR can either be separate from the sources and the KG with references to data artifacts or there can be combined solutions for both the data and their metadata. While there may be several metadata repositories for the different sources and processing steps, a central solution is to be preferred to simplify access to all KG-relevant metadata. Specific implementations of MDRs are CKAN[9], Samply [88], or the DBpedia Databus [89], all using specific vocabularies, standard query languages, and databases to implement their relevant features. Concerning metadata exchange, the Open Archives Protocol for Metadata Harvesting[10] framework also allows aquisition of structured metadata.

Fact-level metadata (or annotations) in the KG can be stored either together with the data items (embedded metadata) or in parallel to the data and referenced using unique IDs (associated metadata) [87]. For example, fact-level metadata can support the selection of values and sub-graphs [90], or the compliance to used licenses in target applications. Such annotations are also useful for other kinds of metadata. Temporal KGs can be realized by temporal annotations to record the validity time interval (period during which a fact was valid) and transaction time (time when a fact was added or changed) [34, 44]. The possible implementations for fact-level annotations depend on the used graph data model (see Section 2.2).

[9] https://ckan.org/
[10] https://www.openarchives.org/OAI/openarchivesprotocol.html

3.3. Ontology Management

Ontology development is the incremental process of creating or extending an ontological knowledge base [91]. KG construction requires to develop an ontology for the initial KG and to incrementally update the ontology to incorporate new kinds of information. As of today, ontology development and curation is still done broadly manual or crowdsourced although some semi-automatic approaches are also proposed. Semi-automatic ontology development tasks share a great overlap with methods from knowledge extraction, entity resolution, quality assurance and knowledge completion. Creating the initial ontology can be derived from a single source that ideally provides already some useful ontology to build on. Public web wikis, catalogs, APIs, or crowdsourced databases are valuable starting sources as they may already contain a large amount of (semi-)structured data on general or domain-specific topics. However, cleaning and enrichment processes are required to ensure sufficient domain coverage and quality to build an initial knowledge graph structure from this existing data. For example, if Wikipedia is used as a primary source its category system can be good start to derive the most relevant classes for the KG by some NLP-based "category cleaning" [9]. Semi-automatic approaches mostly focus on learning an ontology from single sources, i.e. transforming a source into an equivalent ontology or KG. These individual ontologies or KGs can then be integrated into a previous version of the overall KG. A key prerequisite for this kind of ontology integration is the step of ontology and schema matching to determine equivalent ontology and schema elements (classes, properties). After a discussion of semi-automatic approaches for ontology learning we therefore discuss ontology/schema matching and close with approaches for ontology integration.

3.3.1. Ontology Learning

There are two main subfields of ontology learning; the first focuses on learning from from text sources (unstructured data); the second from relational databases (structured data). Although the authors in [92–94] discuss that automatic ontology construction is not likely to be possible, a significant amount of research has been done to support the semi-automatic construction for single sources.

Al-Aswadi et. al. [92] give a state-of-the-art overview of ontology learning from **unstructured text** where the goal is to identify the main concepts and their relations for the entities in a document collection. The approaches [92–94] can be grouped into linguistic approaches (using NLP techniques such as part-of-speech tagging, sentence parsing, syntactic structure analysis, and dependency analysis methods) and machine learning approaches. The latter include statistic-based methods (e.g., utilizing co-occurrence analysis, association rules, and clustering) and logic-based approaches using either inductive logic programming or logical inference. Al-Aswadi et. al. [92] argue that there is a need to move from shallow to deep learning approaches for deeper sentence analysis and improved learning of concepts and relations.

Ma et al. [95] give a survey of methods for learning ontologies from **relational databases** with a focus of methods for reverse engineering or the use of mappings to transform a relational database (schema) into an ontology or knowledge graph. Reverse-engineering allows one to derive an Entity-Relationship diagram or conceptual model from the relational schema. Here additional considerations are needed to deal with trigger and constraint definitions to avoid a semantic loss in the transformation. For the mapping techniques to transform RDBs to KGs, the authors differentiate between template-based, pattern-based, assertion-based, graph-based mapping, and rule-based mapping approaches. The resulting mappings should be executable on instance data in order to generate a graph structure from a relational database [96, 97].

3.3.2. Ontology/schema matching

Consolidating and integrating information from multiple heterogeneous sources requires harmonizing the ontologies and/or schemas of the sources. A main step for such a data integration is ontology and schema matching which is the task of identifying corresponding ontology and schema elements, i.e. matching ontology concepts and matching properties of concepts and entities. For example, to integrate a new source into a KG it is necessary to perform a matching of the source ontology/schema with the KG ontology to identify which source elements are already existing in the KG ontology and which ones should be added. Property matching is also important for entity resolution and entity fusion in order to determine matching entities based on the similarity of equivalent properties and to combine equivalent properties to avoid redundant information. In some cases known entity matches can be

used to aid in the ontology matching step [98]. Some tools also perform entity resolution and ontology matching in combination [99].

There is a huge amount of previous research on schema and ontology matching, although mostly outside the context of knowledge graphs, and there are numerous surveys and books about the topic [100–104]. The match approaches typically rely on determining the similarity of elements using different strategies, such as the similarity of concept/property names or the similarity of instance values. Structural information can also be beneficial in the matching process, e.g. by looking at the concepts in the graph neighborhood. Matching systems commonly rely on a combination of different match strategies in order to achieve high-level match-quality [105].

While string similarity can be a strong signal for a match decision, often semantically similar words are used, which are dissimilar on character level. Dictionaries or pre-trained word embeddings are therefore helpful to capture this semantic similarity. Zhang et. al. [106] investigate how word embeddings (using Word2Vec [107]) can be used for the task of ontology matching. They found a hybrid approach to perform the best, which takes the maximum of either edit-distance-based or word embedding similarity for each entity pair. Another approach, which relies on word embeddings but also on meta-information about property's names and their values as input for a dense neural network is LEAPME [108]. This system trains a classifier based on already labeled property pairs. This trained model can then decide whether unlabeled property pairs and their similarity scores constitute a match. Graph embeddings have also seen some attention in ontology matching as they can capture structural information of an ontology. For example Portisch et. al. [109] use a variation of RDF2Vec [110], which is a walk-based embedding technique similar to Word2Vec, to encode both ontologies and then use a rotation matrix to align the embeddings.

There are some tailored ontology matching approaches for KGs such as for mapping categories derived from Wikipedia to the Wordnet taxonomy with the goal to achieve an enriched KG ontology [9]. Other specific approaches have been developed to address the integration of RDBs with ontologies, that go beyond the mapping languages described in Section 3.1.3. KARMA [111] provides a semi-automatic approach to link a structured source such as a RDB with an existing ontology. The process consists of assigning semantic types to each column, constructing a graph of all possible mappings between the source and the ontology, refining the model based on user input and finally generating a formal specification of the source model.

3.3.3. Ontology Integration

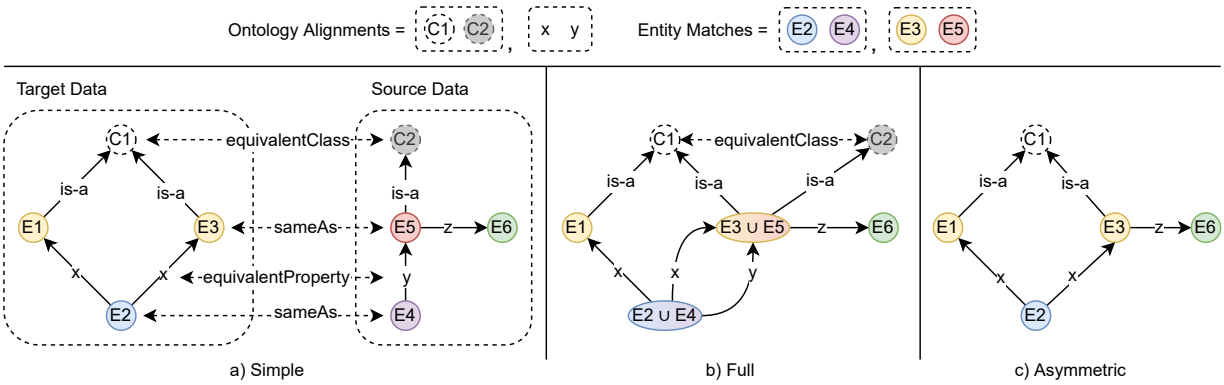

Fig. 3. Ontology and Entity Merging Strategies.

Merging new ontology or schema data into the existing KG ontology is a sub-task of ontology or schema integration. This topic has achieved some attention where recent approaches utilize the mapping result of a match (alignments) to combine multiple ontologies/schemas [112–114]. If the match mapping is automatically determined, it must first be manually validated and possibly corrected to provide a valid basis for the merge step. Osman et al. [114] give a comprehensive and recent summary of ontology integration techniques, which can handle the merging of ontology and entity data using respective alignments. The authors distinguish the following merging strategies:

- **Simple Merge.** Imports all input ontologies into a new ontology and adds bridging constructs between equivalent entities, like defining OWL *equivalentClass* or *equivalentProperty* relations.
- **Full Merge.** Imports all source ontologies into a new ontology and merges each cluster of equivalent entities into a new unique entity with a union of all their relations and leaving equivalent classes untouched.
- **Asymmetric Merge.** These approaches import source ontologies into a preferred target ontology, preserving all its concepts, relations, and rules by merging matching entities into existing target entities or else by creating new ones.

Figure 3 visualizes each of the three strategies, where a) also shows the source and target data that are merged. From the three merging strategies, the authors favor the last and mention it as a good solution for incremental ontology integration. The reason for this preference is that the asymmetric merge strategy preserves the target ontology during the integration, and only adds new elements from a source ontology if necessary. This can also be seen, in Figure 3 c), where the target data is left unchanged, and only a new entity `E6` is added, which is connected to `E3` from the target data. An example approach for asymmetric ontology merging focusing on is-a relations is proposed in [113].

3.4. Knowledge Extraction

Knowledge extraction is a process to obtain structured, more computer-readable data from unstructured data such as texts or semi-structured data, like web pages and other markup formats. The extraction methods of semi-structured data often use a combination of data cleaning (Section 3.1.4) and rule-based mappings (Section 3.1.3) to transform the input data into the final KG, targeting already defined classes and relations of the existing ontology. Most of the work focuses on knowledge extraction from text, sometimes additionally considering images and figures within the text. Recently, there has been an increased interest in creating multi-modal knowledge graphs (i.e. KGs with not only text but also other modes of data such as images) necessitating appropriate methods of knowledge extraction. The detailed discussion of such methods lies outside the scope this paper, but we refer the interested reader to the following survey by Zhu et. al. [10]. The main steps of text-based knowledge representation are named-entity recognition, entity linking, and relation extraction. These steps are discussed in the following and allow the extraction of entities and relations from text for inclusion into a KG. An example of this process is shown in Figure 4.

3.4.1. Named Entity Recognition

Named-entity recognition (NER) refers to demarcating the locations of entity mentions in an input text. In the most widely used scenarios mentions of only a handful of types (e.g. persons, places, locations, etc.) are determined. However KGs usually contain hundreds or thousands of types. Furthermore, off-the-shelf NER tools do not provide canonicalized identifiers for the extracted mentions. A second step is therefore necessary to link entity mentions either to existing entities in a KG or with new identifiers.

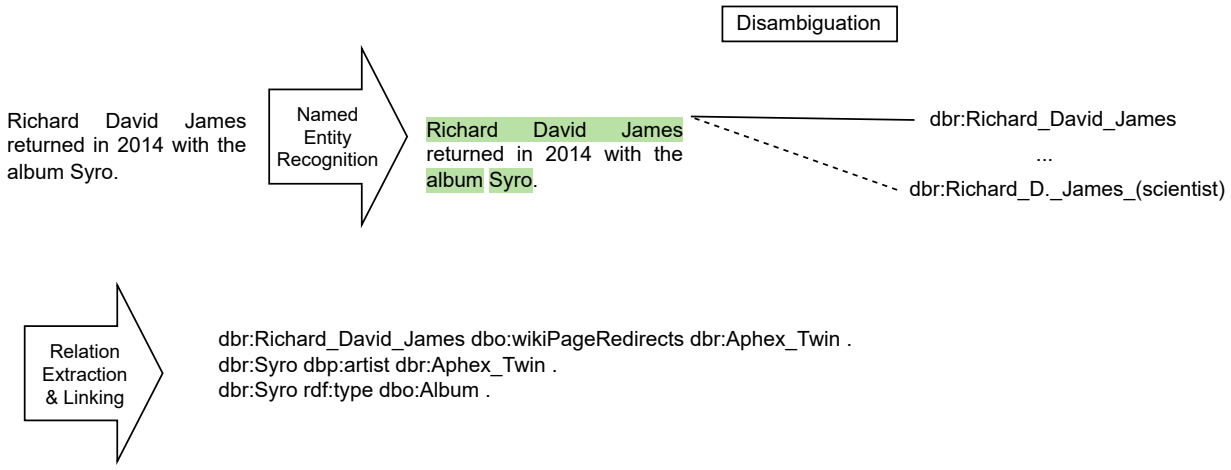

Fig. 4. Knowledge Extraction steps for an example sentence linking entities and relations to the DBpedia KG.

A relatively reliable and simple way to detect entity mentions in a text is the use of a *dictionary* (also referred to as *lexicon* or *gazetter*), which maps labels of desired entities to identifiers in the KG. In addition to it's simplicity such an approach already provides recognized entities in a text with the right link to the KG (i.e. solving the tasks of named-entity recognition and entity linking in one step). However these dictionaries are usually incomplete. A simple way to increase the coverage of such dictionaries is to utilize disambiguated aliases in high-quality sources [9]. Wikipedia redirects or DBpedia's `dbo:alias` property would be a simple way to enhance an entity dictionary. For example https://en.wikipedia.org/wiki/Richard_D_James from the example redirects to https://en.wikipedia.org/wiki/Aphex_Twin. To make dictionary lookups efficient different data structures have been proposed. For example prefix tries or inverted indexing have shown to be a scalable solution for large Web search engines and are used in the NER approaches AGDISTIS [115], TagME [116] and WAT[117].

Machine learning methods have become increasingly popular to tackle NER. This is especially useful to find "emerging entities", i.e. entities that are unknown to the knowledge base. The machine learning models for entity recognition generally fall into the task of *sequence labeling*. A widely successful method for this task is known as Conditional Random Fields (CRF), which uses an undirected graph connecting input and output variables and models the conditional probability of output given the input. Generally these graphs form a linear chain (e.g in the Stanford CoreNLP package [118]), which means for a prediction only the immediate neighbors are relevant in a sequence. While CRFs require extensive feature engineering Deep Neural Networks have become highly popular in recent years for the task of NER, since they do not necessitate this amount of human interaction. For example *LSTM* networks ("long short term memory"), which are a specific case of *recurrent neural networks* (RNN), have become a prevalent choice for NER tasks [119]. The memory cells contained in this architecture are able to deal with long-term dependencies, which was previously a major painpoint for RNNs. Deep learning-based approaches for NER are surveyed in [120].

As multi-modal data become increasingly popular, e.g. on social media platforms, several recent studies focus on multi-modal NER (MNER) [121, 122], where the goal is to leverage the associated images to better identify the named entities contained in the text. Furthermore, there are first approaches [123–125] that address MNER for KGs. They aim to correlate visual content with textual facts. One typical solution parses images and texts to structured representations first and grounds events/entities across modalities. However, intra-modal relation extraction and cross-modal entity linking still are largely unresolved problems.

3.4.2. Linking

If named entities are recognized in a text they need to be linked to the knowledge base or KG. This is called *entity linking* (EL) or *named entity disambiguation* (NED). Given a set of candidates from the knowledge base an EL algorithm needs to decide which entity a mention belongs to. In Figure 4 this can be seen, where `Richard David James` is linked to the DBpedia entity `dbr:Richard_David_James`.

EL algorithms can rely on a variety of features. Based on the mention itself the *confidence* of the used NER tool can be used, how *similar* the mention and the entity are or how much *overlap* exists across mentions [126]. The context of the extracted mentions can provide useful as well. Keyword-based similarity can be used by relying on TF-IDF scores, where rare keywords used in the mention's context, which are connected to a candidate entity can give hints for linkage [127, 128]. Words that occur frequently in the same context can also help in the disambiguation process. Here pre-trained word embeddings can prove especially useful, since they encode semantic similarity in a latent space. Furthermore, already disambiguated mentions can be used to aid in the linking of entities that occur in the same paragraph.

Holistic entity linking [126, 129] approaches leverage background information in the decision process, that exceeds merely using the similarity between mention and entity. Popularly, the graph-structure of Wikipedia links can be used to determine *commonness* and *relatedness*. Commonness refers to the probability that an entity mention links to the respective Wikipedia article of the given candidate entity. Relatedness measures how many articles in Wikipedia link to articles of both candidates. Using such background knowledge unambiguous mentions can aid in the correct linkage of ambiguous mentions [130].

Entity linking approaches furthermore need to address specific challenges such as *coreference resolution*, where entities are not consistently referred to by their names, but with indirect references such as pronouns [126] and how to deal with *emerging entities*, i.e. entities that are recognized, but not yet existing in the target KG. For example

Hoffart et. al [131] keep a contextual profile of emerging entities and when this profile contains enough information to infer the semantic type of the mention it can be added to the KG with it's type.

Generally entity linking and the later discussed entity resolution (Section 3.5) share similarities in aiming to connect the same entities in and across data sources. entity linking and entity resolution are sometimes jointly discussed under the term *entity canonicalization* [9]. While entity resolution typically deals with at least semi-structured data sources, there have been some efforts to address cases with unstructured sources, where deep learning-based approaches are advantageous [132]. However, there are some key differences not only in the characteristic modality of the data sources, but for example in the signals that lead to a linking decision. For example in entity linking already linked mentions of an entity x make it more likely that close encounters with a similar mention also lead to entity x, however in entity resolution in many cases the linkage scenario more closely follows a $1-1$ matching between two data sources under the assumption of deduplicated or clean data sources. It might be worthwhile however to investigate how well entity resolution approaches for dirty sources where multiple entities may match with the same KG entity can be utilized for entity linking and vice versa.

3.4.3. Relation Extraction

Given the identified entities in a text, relation extraction aims to determine the relationship among those entities. In Figure 4 we see this for example, when the text snippet `album Syro` becomes the triple `dbr:Syro rdf:type dbo:Album`, i.e. the type relation for entity `dbr:Syro` is determined.

The first techniques used handcrafted patterns to extract relations. In order to improve coverage different ways to enhance such patterns were devised. The human involvement in these techniques however is a limiting factor. To address these shortcomings statistical relation extraction models were devised. Feature-based methods rely on lexical, syntactic and semantic features to use as input for relation classifiers. Similarly, kernel-based methods [133] rely on specifically designed kernel functions for SVMs to measure the similarity between relation candidates and text fragments. Graph-based methods further integrate known relations between entities and text in order to correctly identify relations [9].

While such methods can be incredibly useful to obtain relatively simple relations with high accuracy they are limited in terms of their recall or at least require a high degree of additional human involvement for feature engineering, designing of kernel functions or the discovery of relational patterns. Neural relation extraction methods aim to close this gap. The input text is transformed via (pre-trained) word embeddings and position embeddings into a format that is suitable for the neural networks that are trained for relation extraction. Instead of devising hand-crafted features the focus in this area lies on investigating various neural network architectures such as recurrent neural networks, convolutional neural networks and LSTMs. The bottleneck for these approaches lies in the availability of training data. A common approach to address this is via distant supervision. Statements from a given data source (for example Wikipedia) are used to train the given model. An overview over the capabilities and challenges of these approaches can be found in [134].

A special case of relation extraction aims to extract relations freely without a pre-defined set of relations. This is known as Open Information Extraction (OpenIE). While this can be a good way to increase the variety of information contained in the KG, a secondary step is necessary to *canonicalize* the extracted relations in order to deduplicate and possibly even link them to already contained synonymous relations in the KG [135].

Several tools exist for the entire process of Knowledge Extraction, with some tools focusing on specific aspects. For example DBpedia Spotlight [136] mainly aims at performing named entity extraction and links those mentions to the DBpedia KG. The dstlr [137] tool extracts mentions and relations from text, links those to Wikidata and furthermore populates the resulting KG with more facts from Wikidata. OpenNRE [138] provides an extensible framework for neural relation extraction, with trainable models, however this approach would necessitate an independent linking step afterwards.

Analogously to NER there are also efforts to use images as information sources for relation extraction. These can range from rule-based approaches [139], which for example verbalize detected spatial relations of recognized entities in an image, to learning-based techniques, which encode visual features of detected objects as well as textual features into distributed vectors used to predict relations between given objects. For example MEGA [140] aligns information contained in the syntax tree and word embeddings of the textual data and the scene graph obtained

from the image. A scene graph connects detected objects in an image via their visual relations. After the alignment process the respective representations are concatenated and sent to a Multilayer Perceptron to predict the relation.

3.5. Entity Resolution and Fusion

Entity resolution (ER), also called entity matching, deduplication or link discovery, is a key step in data integration and for good data quality. It refers to the task of identifying entities either in one source or different sources that represent the same real-word object, e.g., a certain customer or product. An enormous amount of research has dealt with the topic as evidenced by numerous surveys and books [141–146]. In addition to several research prototypes there are also many commercial solutions such as IBM's InfoSphere Identity Insight[11] or SAP's Master Data Governance Platform[12]. Most known approaches tackle static or batch-like entity resolution where matches are determined within or between datasets of a fixed size. The more recent of these approaches deal with multi-source big data entity resolution [147, 148], rely on Deep Learning [132, 149] or KG embeddings [150, 151], with the neural methods having seen more scrutiny recently after an era of relative hype [152].

For KG construction, however, we need incremental approaches that build on previous match decisions and determine for new entities if they are already represented in the KG or whether they should be added as new entities. Furthermore, for streaming-like data ingestion into a KG a dynamic (real-time) matching of new entities with the existing KG entities should be supported. Entity resolution results are fed to the step of *entity fusion* which fuses together matching entities to combine and thus to enrich the information about an entity in an uniform way.

In the following we first discuss proposed approaches for incremental ER and then for entity fusion.

3.5.1. Incremental Entity Resolution

Entity resolution is challenging due to the often limited quality and high heterogeneity of different entities. It is also computationally expensive because the number of comparisons between entities typically grows quadratically with the total number of entities. The standard approach for entity resolution uses a pipeline of three succeeding phases called blocking, linking/matching and clustering [58, 153]. The main step is to determine the similarity between pairs of entities to determine candidates for matching. This matching step often results in a similarity graph where nodes represent entities and edges link similar pairs of entities. The preceding blocking phase aims at drastically reducing the number of entity pairs to evaluate, e.g. based on some partitioning so that only entities in the same partition need to be compared with each other (e.g., persons with the same birth year or products of the same manufacturer). After the match phase there is an optional clustering phase that uses the similarity graph to group together all matches. This clustering can typically improve the quality of entity resolution and also assists the succeeding step of entity fusion to fuse the matching entities into one representative entity for the KG.

For incremental ER the task is to match sets of new entities from one or several sources with the current version of the KG which is typically very large and contains entities of different types. It is thus beneficial to know the type of new entities from previous steps in the KG construction pipeline so that only KG entities of the same or related types need to be considered. Figure 5 illustrates a high level workflow for incremental ER. The input is now the current version of the KG with the already integrated entities (previous clusters in Figure 5) as well as the set of new entities to be integrated. This requires the development of incremental versions for blocking, matching and clustering phases that focus on the new entities. To allow better match decisions for incremental ER it is generally advantageous to retain the entities of the previously determined clusters (and their match similarities) and not only the fused cluster representatives. Some incremental clustering schemes can also use this to identify previous match mistakes and to repair existing clusters for new entities [58, 154]. Some approaches [155] and tools [156] also support to execute incremental ER in parallel on multiple machines to improve execution times and scalability to deal with large KGs.

Blocking for incremental or streaming ER requires to identify for the new entities all other entities in the KG that need to be considered for matching. Given the typically high and growing size of the KG it is important to limit the matching to as few candidates as possible and determining the candidates should also be fast. As mentioned, blocking and entity resolution should be limited to entities of the same (or most similar) entity type and one can

[11] https://www.ibm.com/products/infosphere-identity-insight
[12] https://www.sap.com/products/technology-platform/master-data-governance.html

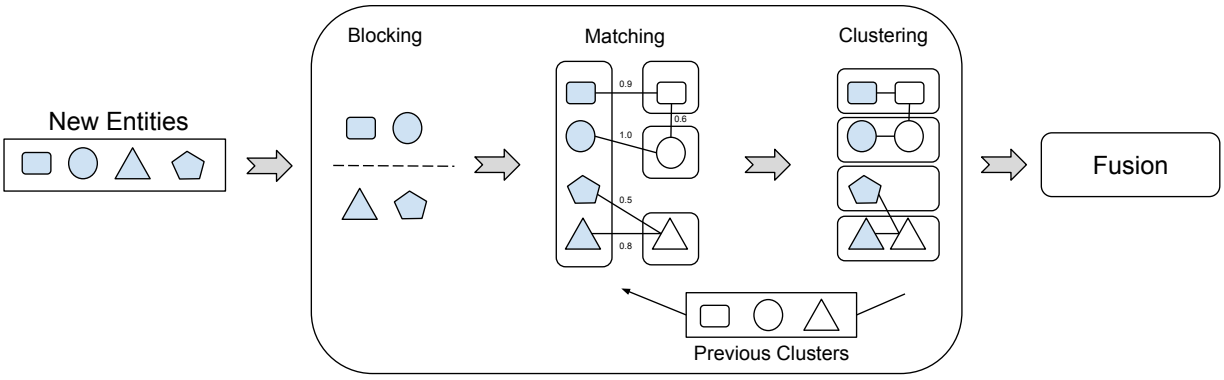

Fig. 5. Incremental entity resolution workflow.

apply the same blocking approach for the new entities as for the previously integrated entities, e.g. by using some attribute-based blocking key such as birth year for persons or manufacturer for products. Several works [145, 146] have proposed further improvements over such a base approach for specific cases and blocking approaches. One approach [157–159] is to keep the blocking keys in a data structure with efficient data access to make comparisons among entities faster. Another method to speed up the computation is the use of so-called summarization techniques [159]. One such approach summarizes (divides) larger blocks into multiple sub-blocks with a representative and directing a new record (query) to the sub-block with the most similar representative. This enables a constant number of comparisons for each new record which is valuable for both incremental and streaming ER. While most blocking approaches rely on domain or schema knowledge, there are also so-called schema-agnostic blocking schemes for highly heterogeneous data where entities of a certain type can have different sets of attributes. Hence, schema-agnostic approaches consider most or all attribute values and their components (e.g. words or tokens), regardless of the associated attribute names. While there are many schema-agnostic blocking approaches for non-incremental ER [145, 146] schema-agnostic blocking approaches for incremental or streaming ER have only recently been proposed [155, 160, 161].

The *matching* step of incremental ER is limited to the new entities and involves a pair-wise comparison with the existing KG entities determined by the preceding incremental blocking step. The main goal is to determine all similar entities as potential match candidates as input for the final clustering step, where it is decided whether the new entity is added to an existing cluster or whether it should form a new cluster. Pairwise matching can be done as for batch-like ER and is based on the combined similarity between two entities derived from property values or related entities. The match approach can be configured manually, e.g., together with some similarity threshold that should be exceeded for match candidates, or by applying a supervised machine learning model [145]. If the pairwise match relationships between previously integrated KG entities are maintained in a similarity graph spawning the previous clusters, this graph can be extended by the new entities and links to the newly determined mach candidates as an input for incremental clustering [156].

While there are many approaches for batch-like *entity clustering* [162, 163], the incremental maintenance of entity clusters for new entities has received comparatively little attention. A straight-forward approach is to simply add a new entity either to the most similar existing cluster or to create a new cluster if there is no previous cluster with a high enough similarity exceeding some predefined similarity threshold [164]. However, this approach typically suffers from a strong dependency on the order in which new entities are added. In particular, wrong cluster decisions, e.g., due to data quality problems, will not be corrected and can lead to further errors when new entities are added. A more sophisticated incremental approach based on correlation clustering is proposed in [154] that maintains previous clusters within a similarity graph. The updated similarity graph is not only used to determine the clusters for new entities but to also repair previous clusters, e.g. by splitting and merging clusters or by moving entities among clusters. The incremental approaches in [57, 58] support optimized clustering decisions for duplicate-free (sometimes called clean) data sources from which at most one entity can participate per cluster of matching entities. In this case, an effective clustering strategy is the so-called "max-both" approach where an entity s from a set of

new entities is only then added to the most similar cluster c when there is no other new entity that is more similar to c than s. The approach of [58] also supports a light-weight cluster repair called n-depth reclustering where only entities close to new entities in the updated similarity graph are considered for changing clusters.

3.5.2. Entity Fusion

Merging multiple records of the same real-world entity into a single, consistent, and clean representation is referred to as data fusion [165]. This is a main step in data integration as it combines information from several entities into one enriched entity. Data fusion still entails resolving inconsistencies in the data. First the records may disagree on the names of matching attributes so that one preferred name has to be chosen that should be the consistent with the attribute names of other entities of the same type to facilitate querying. Furthermore, the matching records can disagree on the values of an attribute. There are three main strategies to handle such attribute-level inconsistencies or conflicts [165]:

- Conflict Ignorance: The conflict is not handled but the different attribute values may be retained or the problem can be delegated to the user application.
- Conflict Avoidance: It applies a unique strategy for all data. For example, it prioritizes data from trusted sources over others.
- Conflict Resolution: It considers all data and metadata before applying a decision to apply a specified strategy, such as taking the most frequent, most recent or a randomly selected value.

Such techniques were first applied for relational data but also found use for Linked Data fusion [166]. A valuable strategy is to combine multiple value scoring functions. Mendes et al. [167] combine two methods named *TrustYourFriends* (prioritizing data from the trusted source) and *KeepUpToDate* (using the latest value) for conflict avoidance and resolution. Moreover, they apply input quality assessment metrics to filter out the values below a threshold or keep the values with the highest quality assessment. Other techniques such as computing average, minimum, and maximum or taking the most frequent values are provided by their data integration framework. Dong et al. [168] combine *TrustYourFriends* with a *Weighted Voting* (most frequent or similar values) approach, whereas the former source ranking score is calculated based on an statistical approach. Similarly, Frey et al. [90] apply a median-based approach for Linked Data fusion. They distinguish functional properties[13] and assign a single value to them. Non-functional properties can be assigned multiple different values.

3.6. Quality Assurance

The quality of a KG is crucial for its credibility and therefore its usability in applications [169]. Quality assurance is the task to maintain a high KG quality despite the continuous evolution of the KG. It comprises quality evaluation as well as quality improvement **including knowledge completion** (Section 3.7). Quality evaluation serves to assess the quality and detect quality issues. Quality improvement aims at fixing or mitigating the detected quality issues by refining and repairing the KG. This encompasses inferring and adding missing knowledge to the graph, or identifying and repairing erroneous pieces of information in order to improve data quality. Quality evaluation is not only important for the resulting KG as outcome of the KG construction process but also within the different construction tasks, such as data cleaning for acquired data, knowledge extraction, ontology evolution or entity fusion. The data cleaning approaches mentioned in Section 3.1.4 can also be applied to the KG, e.g. to identify outliers or contradicting information. Metadata such as provenance information is also important for quality assurance, for example to explain and maintain KG data concerning the context and validity of conflicting values [9].

Different methods for **quality evaluation** have been proposed that can be categorized into internal and external methods [15]. *Internal methods* only use the current KG for evaluation. One conventional approach is to evaluate the accuracy of the KG against a manually labeled subset of entities and relations [15]. However, this is costly, so that those gold standards are usually small. Other approaches use statistical methods such as distance-based, deviations-based and distribution based methods [170].

By contrast, *external methods* also use external knowledge sources for quality evaluation such as crowdsourcing methods or other KGs.

[13] A functional property is specified to have a maximum cardinality of one. e.g., a person entity should only have one value for date of birth.

- **Crowdsourcing approaches**: Acosta et al. [171] leverages the wisdom of the crowds in two ways. They launched a contest targeting an expert crowd in order to find and classify erroneous RDF triples, and then published the outcome of this contest as paid microtasks on Amazon Mechanical Turk (MTurk) in order to verify the issues spotted by the experts. Their empirical evaluation on DBpedia shows that the two styles of crowdsourcing are complementary, and that crowdsourcing-enabled quality assessment is a promising and affordable way to enhance data quality.
  Paulheim et al. [15] define *retrospective evaluation* as a method in which the human judges the correctness of the KG The reported quality metric is accuracy or precision. Since KGs are often voluminous, the retrospective approach is typically restricted to a KG sample.
- **Using external knowledge bases**: Li et al. [172] investigate the correctness of a fact by searching for evidences in other knowledge bases, web data, and searching logs. Similarly, [173] suggests individual checking of a single facts in different datasets in order to detect inaccurate facts. This is also useful for yielding larger-scale gold standards, but has two sources of errors: errors in the target knowledge graph, and errors in the linkage between the two.

3.7. Knowledge Completion

Knowledge Graph completion is the task of adding new entries (nodes, relations, properties) to the graph using existing relations. Paulheim [15] surveys KG completion approaches as well as evaluation methods. He also distinguishes internal from external methods, especially for determining missing entity type information and relations. The survey concludes that current approaches for KG completion typically limit themselves to a single task such as determining missing type information, or missing relations (link prediction) or missing attribute values (literals). Holistic solutions to simultaneously improve the quality of KGs in several areas are thus currently missing.

3.7.1. Type completion

Type completion refers to the task of assigning types to nodes without type information. In the case of PGM it is in most cases not allowed to have nodes without type information [28]. Since there is no universally agreed upon definition of the PGM, and the possibility to label nodes with unknown type with a dummy label, type completion can still be seen as a relevant KG completion task. In this case node classification approaches can be used to predict classes of unlabeled nodes. For example Neo4j provides a specific node classification pipeline[14], although the resulting predictions are added as node properties necessitating a post-processing step to redefine the label.

The traditional way of determining missing type information in RDF datasets involves the use of logical reasoning. However this approach is limited since it relies on already consistent facts and existing `rdf:type` information in the knowledge base [174]. To address this shortcoming statistical approaches use the distribution of relations between entities to predict missing type information. For example SDType [175] uses a weighted voting approach based on the statistical distribution of the subject and object types of properties. Similarly StaTIX [176] relies on weighted statistics of multiple properties of entities as input for their clustering approach.

Recently, there has been some attention on leveraging KG embeddings to infer type information. For example ConnectE [177] incorporates two mechanisms with one relying on *local typing knowledge* and the other on *global triple knowledge*. The first relies on the fact that entities close in the embedding probably share the same type. Relying on relationship information the second mechanism learns entity type embeddings by replacing the head and tail entity in a triple for their corresponding type. Finally, for entity type prediction a composite score of the two mechanisms is used.

3.7.2. Link Prediction

The task of link prediction aims at finding missing relations in a KG. In the example presented in Figure 1 we see that while there is a *writtenBy* relation between the song *Xtal* and the artist *Aphex Twin*, there is no relation between the song *Ageispolis* and *Aphex Twin*. Predicting this missing *writtenBy* relation would be a goal of link prediction.

[14] https://neo4j.com/docs/graph-data-science/current/machine-learning/node-property-prediction/nodeclassification-pipelines/node-classification/

Based on [15], a common method for the prediction of a relation between two entities is distant supervision [178–181] using external resources. This method starts with linking entities of the Knowledge Graph to the text corpus using NLP approaches and then tries finding patterns in the text between entities. Another approach [182] uses the same methodology, but considers the whole Web as the corpus. Lange et al. [183] learn patterns on Wikipedia abstracts using Conditional Random Fields [184]. Blevins et al. [185] propose a similar approach, but on entire Wikipedia articles. Another line of research uses semi-structured data such as tables [186, 187] or list pages [188] in Wikidata for predicting missing relations.

In the last years considerable research attention was devoted to investigating KG embeddings for the task of link prediction. These methods encode entities and relations of a KG as low-dimensional vectors in an embedding space. The existing triples in a Knowledge Graph can be used to train such models, evaluating their performance on a held-out set of triples. For example TransE [189] encodes relations as translations from head to tail entity of a triple $(\textit{head}, \textit{relation}, \textit{tail})$. This is done by minimizing the distance between `h` + `r` and `t`, where `h`, `r` and `t` are the embeddings of *head*, *relation* and *tail* respectively. A variety of approaches have been devised to address the problems of TransE to model $1-n$ or $n-n$ relations, by e.g. encoding relations in a separate hyperplane [190] or operating in the hyperbolic space [191]. For a more broad overview and benchmark study we refer to this paper [192].

The described embedding-based link prediction methods rely on *shallow embeddings*, which means all embeddings are stored in a entity/relation-matrix and obtaining the respective embedding for an entity or relation is done by using a lookup-table. These approaches are unable to deal with unseen entities. The study of *inductive* link prediction aims to address this shortcoming. GraIL [193] relies on Graph Neural Networks (GNN) to achieve this. This approach samples the subgraph enclosing the link to be predicted, then labels the nodes in this subgraph based on the distance to the target nodes (i.e. the nodes which the link would connect). The labeled subgraph is then used in a GNN to score the likelihood of a triple. NodePiece [194] can perform inductive link prediction via a compositional representation for entities. Relations around a node are sampled in order to create a node hash, which is passed through an encoder to obtain the final entity embedding. Being able to create entity representation for unseen entities, but known relations permits NodePiece to then use any scoring function (e.g. TransE) for the link prediction task.

A special type of link prediction aims to discover identity links (e.g. `owl:sameAs` relations) [7], which connect nodes, that refer to the same entity. This task serves the same goal as entity resolution (discussed in Section 3.5).

3.7.3. Data Enrichment & Polishingh

Concerning aspects of domain coverage and succinctness, additional processes are applicable that increase the final quality of the KG. In addition to type and relation prediction, domain knowledge could possibly be extended by loading completing entity information from external (open accessible) knowledge bases. This approach is different to the process of integrating an entire external data collection but only focusing on loading necessary domain information that relates to the already integrated entities. For enhancing KG data with additional relevant domain entities information external knowledge bases can be requested based on extracted (global) persistent identifiers (PID). For example extracted ISBN numbers, DOIs, or ORCIDs allow to request additional external information from Wikidata; or Gene and Protein data is accessible based on their symbols in public biochemical databases, like the National Library of Medicine[15]. Paulheim surveys approaches that exploit links to other KGs in order to not only verify information but also to find additional information to fill existing gaps [15].

In contrast to data enrichment, it might be desirable to filter out unnecessary entities that do not belong to the targeted domain, only bloating the KG. Applying automatic approaches can cause extraction of irrelevant information, and requires techniques either of manual nature or by leveraging known information from external already structured databases.

4. Overview of Knowledge Graph Construction Pipelines and Toolsets

We now investigate and compare construction pipelines for existing KGs and for KG construction toolsets with respect to the requirements and construction steps introduced in the previous sections. The KG-specific approaches

[15] https://www.ncbi.nlm.nih.gov/

Table 3
Overview of selected KGs. '*' in the first column indicates manually curated (crowd-sourced) KGs. '?' means unknown/undisclosed values. The statistics include the KG's year of announcement, targeted domain, processed number of data sources, KG data model, graph size, number of versions, and year of last update. Used domain abbreviations: Cross = cross domain, MLang = multi-lingual data.

	Year	Domain	Srcs.	Model	Entities	Relations	Types	R-Types	Vers.	Update
Closed KG										
Google KG [195]	2012	Cross,MLang	>>>1	Custom,RDF	1B	>100B	?	?	?	?
Diffbot.com	2019	Cross	>>>1	RDF	5.9B	>1T	?	?	?	?
Amazon PG [196]	2020	Products	>1	Custom	30M	1B	19K	1K	?	?
Open Access KG										
*Freebase [197]	2007	Cross	>>1	RDF	22M	3.2B	53K	70K	>1	2016
DBpedia [198]	2007	Cross,MLang	140	RDF	50M	21B	1.3K	55K	>20	2023
YAGO [199, 200]	2007	Cross	2-3	RDF(-Star)	67M	2B	10K	157	5	2020
NELL [201]	2010	Cross	⩾1	Custom,RDF	2M	2.8M	1.2K	834	>1100	2018
*Wikidata [202]	2012	Cross,MLang	>>>1	RDB/RDF	100M	14B	300K	10.3K	>100	2023
DBpedia-EN Live [203]	2012	Cross	1	RDF	7.6M	1.1B	800	1.3K	>>>1	2023
Artist-KG [204]	2016	Artists	4	Custom	161K	15M	>1	18	1	2016
*ORKG [205]	2019	Research	>>1	RDF	130K	870K	1.3K	6.3K	>1	2023
AI-KG [206]	2020	AI Science	3	RDF	820K	1.2M	5	27	2	2020
CovidGraph [207]	2020	COVID-19	17	PGM	36M	59M	128	171	>1	2020
DRKG [208]	2020	BioMedicine	>7	CSV	97K	5.8M	17	107	1	2020
VisualSem [209]	2020	Cross,MLang	2	Custom	90k	1.5M	(49K)	13	2	2020
WorldKG [210]	2021	Geographic	1	RDF	113M	829M	1176	1820	1	2021

focus on integrating data from a rather fixed set of data sources for a single KG while the toolsets (or strategies) are more generic and can be applied for different sources and KGs. Overall we consider 16 KG-specific approaches (with a focus on ten semi-automatic and open implementations) and seven toolsets. In the first subsection we give an overview of the different approaches including data statistics for the respective KGs and characteristics about the data sources and their construction pipelines. This overview aims already at providing a good assessment of the current state of the art. In the two further subsections we give additional details about the KG-specific approaches and the toolsets. In Section 5 we will discuss remaining challenges and thus areas for future work.

4.1. Overview and Comparison

Our overview is divided into two parts. We first summarize general information and statistics for the selected KGs using Table 3 and then investigate KG construction criteria for KG-specific pipelines and KG toolsets using Table 4. Given the enormous and growing number of KGs, we had to restrict ourselves to a small number of efforts. In our selection we try to cover popular KGs such as DBpedia and Yago as well as more current approaches for either a single domain or several domains (cross domain). Most importantly, we focus on KG projects described in peer-reviewed articles and discuss **closed KGs** only briefly as their data is not publicly accessible and the used techniques are not verifiable. Such closed KGs are typically developed and used in companies such as company-specific Enterprise KGs [213] and the KGs of big web and IT companies such as Google [195], Amazon [196], Facebook, Microsoft [214], Tencent, or IBM. However, open and easy to use KG toolsets are still in their infancy. Here we tried to include recently described approaches that have already been applied to create several KGs including those for a specific domain or a single data type. We expect that our criteria for comparison are also useful to evaluate KG-specific and more generic construction approaches not covered in this paper.

Table 3 summarizes general characteristics of the selected KGs which are grouped into *closed* and *open access* KGs and in each group ordered by their year of announcement or first publication. The table also displays the KG's targeted domain, processed number of data sources, underlying data model, graph size (number of entities, relations, entity types and relation types), number of versions and year of last update. Table 3 excludes the toolset projects as these are not restricted to a single KG.

Table 4

Comparison of KG construction approaches w.r.t.*Consumed Data*, generated *Metadata*, and *Performed Construction Tasks*. The construction tasks are rated as *simple/manual* ○ or *sophisticated/semi-automatic* ● . '?' indicates *mentioned but unclear* implementation. Each criterion can cover multiple solutions.

				Consumed Data					(Meta)Data			Performed Construction Tasks							
Name of System	System Version/Year	Open Implementation	Incremental Integration	Unstructured Data	Semi-Structured Data	Structured Data	(Event-)Stream Data	Supplementary Input	Deep Provenance	Temporal Data	Additional Metadata	KG Initialization	Input Cleaning	Ontology Management	Knowledge Extraction	Entity Resolution	Entity/Value Fusion	Quality Assurance	Knowledge Completion
Dataset Specific																			
DBpedia	2019	✓			✓			✓	✓	✓	✓	○	●	○	○			●	○
YAGO4	2020	✓			✓	✓		✓	✓	✓		○	○	●				●	
DBpedia-Live	2012	✓	○		✓		✓	✓	✓	✓		○	●	○	○				
NELL	2018		●	✓	✓			✓	✓	✓		○		●	●			○	
Artist-KG	2016	✓	○		✓	✓						○	○	●		●			
AI-KG	2020		?		✓			✓	✓	✓		○			●			○	
CovidGraph	2020	✓	○	✓	✓	✓		✓	✓	?		○		?	●	○			
DRKG	2020	✓			✓	✓			✓			?			○		○		●
VisualSem	2020	✓		✓	✓	✓		✓				○	●		○				
WorldKG	2021	✓			✓			✓				●	●	●	○			○	
Toolset/Strategy																			
FlexiFusion [90]	2019				✓	✓		✓	✓		✓	○	○				●		
dstlr [137]	2019	✓	?	✓					✓			○			●			○	○
XI [50]	2020		?	✓	✓				?	?	?	○			●			?	
AutoKnow [196]	2020			✓	✓			✓				○	●	●	●				●
HKGB [211]	2020		○		✓			✓		✓		●		●	●	?		○	●
SLOGERT [212]	2021	✓			✓				✓	✓		○			●	?			○
SAGA [47]	2022		●	✓	✓	✓	✓	✓	✓	✓		?	●	○	●	●	●	●	●

The table includes three manually curated projects based on crowd sourcing, namely the well-known Freebase and Wikidata approaches as well as the newer Open Research Knowledge Graph (ORKG). Freebase [197, 215] was one of the first collaboratively built and versioned KGs and, after its shutdown in 2016, became a popular source for building several other KGs including Wikidata [202]. Wikidata allows the annotation of entities by key-value pairs with a validity time, provenance, and other meta information such as references [216]. As a Wikimedia project, full data dump snapshots are released twice a month. The ORKG [205] focuses on publications where manually uploaded papers are automatically enriched with metadata. The platform provides tools to extract information such as tables and figures from publications and to help find and compare similar publications of interest.

Most of the considered KGs are based on RDF while some use a property graph or custom graph data model (fifth column in Table 3). Regarding the covered domains, the selected KGs either integrate sources from different domains (cross-domain) or focus on a single domain such as research, biomedicine or Covid-19. A possible limitation of cross-domain KGs, especially for smaller-sized ones, is that they can miss domain-specific details or expert knowledge. Some of the KGs contain and connect multilingual information (MLang) by providing descriptive entity values in different languages. These translations are mostly taken directly from one of the sources (e.g., Wikipedia or BableNet), instead of generating own translation during the construction process. There are large differences among the KGs regarding the number of integrated source datasets (from 1 to 140) and the size of the KGs in terms of number of entity and relation types and number of entities and relations. With the highest number of sources, DBpedia independently extracts 140 sources (Wikipedias), with equivalent entities being interlinked by the extracted `sameAs` connections contained in the page articles. The closed KGs are by far the largest with up to almost 6 billion

entities and more than a trillion relations (Diffbot.com). Wikidata is the largest open-source KG with about 100 million entities of 300K entity types and 14 billion relations of 300K relation types. The smallest KGs have less than 1 million entities or relations. In general, the open KGs are rather limited in the number and diversity of the data sources while closed approaches such as Google KG aim at integrating information at web scale. Only a few of the KG projects continuously release updated versions of their KG while most projects only released data once or irregularly every few years. This underlines that continuous maintenance of KGs is not yet commonplace. With over 1100 dumps[16] NELL features the highest number of continuously and incrementally generated KG versions.

We now turn to a closer inspection of the KG construction processes of the individual KGs and toolsets. Table 4 summarizes the corresponding information for the ten open access KGs with semi-automatic construction as well as for seven toolsets / strategies for KG construction. We also provide information on the year of the considered version (publication) and indicate whether the approach offers an open implementation. We see that the pipeline/-toolset implementations for two of the open access KGs (NELL, AI-KG) and even five of the seven toolsets are closed-source including the approaches from Amazon (AutoKnow) and Apple (SAGA). The third column in Table 4 indicates to what degree incremental KG updates are supported, i.e., that changes in the data sources or new sources can be integrated without a full recomputation of the KG. We see that most approaches have either no or unknown support for incremental updates. DBpedia and Yago are limited to batch updates with a full recomputation of the KG while others such as DRKG and WorldKG represent one-time efforts without KG updates at all. Other approaches provide simple incremental capabilities. Artist-KG and CovidGraph are able to integrate new sources incrementally, but do not specify ways to ingest changes in the underlying data sources. DBpedia-Live tracks changes automatically in the underlying data sources and integrates them directly whilst skipping expensive quality assurance steps. Two approaches provide sophisticated incremental capabilities. NELL grows a KG via a semi-supervised approach enabling human interaction to avoid accumulation of errors. SAGA is one of the most sophisticated approaches w.r.t incremental integration. It has the ability to ingest changes into a stable KG, which is updated in batches, and also serves a live KG which forgoes some quality assurance steps in lieu of data freshness. For some approaches their incremental integration capabilities are unclear. AI-KG describe such capabilities as future work and dstlr mention the ability to track document changes via Apache solr, but do not mention any ways to deal with such changes. In the case of the XI pipeline, this feature's support was considered in the final implementations.

In the following, we describe the further criteria considered criteria that fall into three groups regarding the consumed input data, generated metadata and the performed construction tasks.

Consumed Input. For the input data we differentiate the supported kind of data (*unstructured, semi-structured, structured*) and consider support for stream input data and supplementary input data for further processing steps (e.g., metadata, mappings, but excluding tool configurations).

As Table 4 shows, populating KGs from semi-structured data is most common while only about half of the considered solutions or toolset support the import from unstructured or structure data. Several popular KGs (DBpedia, YAGO, NELL) integrate information from Wikipedia and use it as a premium source for a high amount of valuable knowledge. Open accessible databases such as WordNet, ImageNet, or BabelNet are also frequent starting points for KG construction. Only two of the projects support the continuous consumption of event streams (DBpedia Live and SAGA). NELL continuously crawls the web for new data but updates the KG in a batch-like manner. Most approaches integrate supplementary data, especially mapping rules, training data, or quality constraints (SHACL shapes).

Collected Metadata. We consider whether deep or fact-level provenance,temporal information (e.g, validity time) and additional metadata such as aggregated statistics, process reports, or descriptive and administrative information are collected and added to the KG or a separate repository.

The acquisition of provenance data is the most common kind of metadata support and ranges from simple source identifiers and confidence scores up to the inclusion of the original values. Several systems maintain temporal metadata while further metadata is hardly supported or at least not described. In the case of the toolsets, the generation of additional metadata is possible in XI but depends on the use case and resulting pipeline. In general, support for metadata is thus limited and has room for improvement.

[16] http://rtw.ml.cmu.edu/rtw/resources

Construction Tasks. In this group we consider to what degree the eight construction tasks introduced in the previous section are supported.

- **KG Initialization** - Here a common strategy is to manually create the initial KG either by development from scratch or reusing existing KGs. There may also be a complex pipeline to construct the initial KG by processing semi-structured data from catalogs, wikis, or category systems. All projects start with building or using some initial KG data. WorldKG and HKGB semi-automatically build an initial ontology and are therefore more advanced compared to a manual ontology construction.
- **Input Cleaning (Filtering, Correction)** - support for filtering, normalization, or correction of noisy input data. We exclude here NLP/text pre-processing as this is normally part of knowledge extraction. This functionality is not always provided (or documented) and often based on manually defined rules and filter definitions, e.g., to select properties and relationships for certain entity types. Some solutions also apply normalization steps, e.g., to unify date or number representations.
- **Ontology Management** - most approaches have at least some basic (manual) support to evolve the KG ontology and schema data for newly structured input data. In DBpedia, the KG ontology (and data mappings) can be changed manually and needs to be loaded before running a new batch update. The more freshness-oriented approach of DBpedia Live continuously watches ontology changes and immediately schedules affected entities for re-extraction. More advanced approaches rely on a semi-automatic ontology evolution or enrichment. In particular, some systems can identify new entity and relation types in the input data for addition to the ontology after manual confirmation (NELL, HKGB).
  While for example WorldKG relies on an unsupervised ML approach for ontology alignment, most approaches still perform alignment and merging of ontologies manually.
- **Knowledge Extraction** - many solutions use rule-based mappings to extract entities and relations from semi-structured sources (DBpedia, Yago, DRKG, VisualSem, WorldKG). Some tools use machine learning approaches for extraction (AI-KG, CovidGraph, dstlr, SLOGERT), while NELL uses a supervised approach to find new patterns. For entity linking different approaches are used such as dictionary-based approaches relying on gathered synonyms (e.g. AI-KG), use of human interaction (XI), or applying entity resolution (e.g. HKGB). Given the focus on semi-structured data sources, the techniques for knowledge extraction are generally relatively advanced compared to other steps in KG construction. This has also been made possible by the frequent use of existing knowledge extraction tools such as Stanford CoreNLP, as will see in the discussion of the approaches in the next subsections.
- **Entity Resolution** - this task is supported in only by few approaches and the pipelines that do employ ER tend to use sophisticated methods like blocking to address scalability issues (ArtistKG, SAGA), and machine-learning-based matchers (SAGA). HKGB's description of their ER solution is too vague to make a definite statement and for SLOGERT it is mentioned, that in some cases ER might be necessary, but should be done with an external tool. For textual data, identification and matching of entities to KG elements is already covered by entity linking in the knowledge extraction step (Sec. 3.4).
- **Entity Fusion** - this is the least supported task in the considered solutions. None of the dataset specific KGs performs classical (sophisticated) entity fusion in the manner of consolidating possible value candidates and selecting final entity ids or values. Instead, the final KG often contains a union of all extracted values, either with or without provenance, leaving final consolidation/selection to the targeted applications. The DRKG project uses a simple form of entity fusion to normalize entity identifiers.
  Even for the discussed toolsets the coverage of this task is relatively low. The FlexiFusion allows to apply specific fusion functions, leverages provenance information, and performs a stable id assignment for entity and property clusters. SAGA refers to the usage of several truth discovery and source reliability-based fusion methods.
- **Quality Assurance** - human-in-the-loop strategies have been applied to varying degrees, with some solutions, such as HKGB, relying heavily on user interaction. In contrast, others require only final user approval of the correctness of extracted values or patterns, like NELL. The World KG approach manually verifies all matches to the external ontologies. Further, SAGA tries to detect potential errors or vandalism automatically. It quarantines

them for human curation, where changes are treated directly in the live graph and later applied to the stable graph.
Only DBpedia and YAGO perform an automatic consistency check. Additionally, YAGO guarantees ontological consistency by applying a logical reasoner, and DBpedia checks for dataset completeness and measures quality against the former last version.
In our study, only dstlr offers support for validating extracted facts against an external knowledge base.

– **Knowledge Completion** - the integrated KG data is enriched with locally inferred (relations, types) or external knowledge. DBpedia attaches additional entity type information based on current ontology and relation data. Three approaches (DRKG,HKGB,SAGA) presented ML-based link prediction on graph embeddings to find further knowledge. In the case of the DRKG and HKGB approach it is unclear if the newly predicted information flows back into the KG or is stored separately.
Regarding enrichment with external knowledge: While dstlr links entities to Wikidata, it also fetches stored properties from this external source. However, SLOGERT only adds links to external information based on previously extracted identifiers (PIDs).

Overall, we see that the KG-specific approaches have a number of limitations regarding scalability to many sources, support for incremental updates and in several steps regarding metadata, ontology management, entity resolution / fusion, and quality assurance. The toolsets are generally better in terms of their functionality but they are mostly closed-source and thus not usable for new KG projects or research investigations.

4.2. KG Specific Solutions

DBpedia [217] is one of the most popular KGs, establishing a central access point for the Semantic Web. It extracts structured data from Wikipedia article dumps utilizing the DBpedia Extraction Framework (DIEF), which was forked by several other wiki-based KG projects [218]. The DIEF executes numerous extractors, each extracting a specific aspect of the article page, like type information based on the used info-box template or the pages abstract paragraph. One specific extractor, maps semi-structured information from Wikipedia infoboxes[17] to the DBpedia ontology. DBpedia's ontology and infobox mappings are manually curated by the DBpedia community using a publicly accessible mapping wiki[18]. For each extraction run, the latest version of the ontology and mappings are fetched from the wiki API. In the post-processing phase, a type consistency step checks whether an extracted entity and its relations are violating the is-a or has-a definitions of the DBpedia ontology (e.g., a person entity should not have mechanical doors and hence be a car simultaneously). And a completion phase materializes transitive type (is-a) relations for each entity.

Since 2020, the current extraction cycle [198] is built around the Databus [89] (meta)data platform, which allows managing data releases, including descriptions, versioning, data quality reports, and automatic metadata generation. The monthly performed batch extraction consumes data from up to 140 wikis, including several language-specific Wikipedia versions, Wikidata and Wikimedia Commons. Each release is checked for dataset completeness and validated concerning quality. In the final data, equal entities of the different wikis are connected by `sameAs` links derived from the *interwiki links* contained in the data sources.

A testing library helps debugging data problems based on SHACL and other integrity tests. Deep provenance is supported by linking each extracted value with the revision id of the originating article and the applied extractor.

In addition to DBpedia's dump extraction, **DBpedia Live** [219] is a service that provides a real-time KG by performing continuous extraction of changed Wikipedia contents. Changed articles are fetched and reprocessed to extract all relevant values and override (or add/delete) them in the KG. The live extraction also monitors ontology (mapping wiki) changes and schedules all affected pages for re-extraction [203]. For improved performance, the live extraction is not applying the mentioned DBpedia post-processing or quality assurance that are thus only performed when the data is completely reprocessed.

[17] An infobox is a fixed-format table (usually in the top right-hand corner) to consistently present some unifying aspect that articles share.
[18] http://mappings.dbpedia.org

YAGO [199]. The *Yet Another Great Ontology* project initially extracted information about entities of Wikipedia and combined them with an ontology derived from the hierarchically structured WordNet [220]. In version 2, the KG was extended by temporal information (Wikipedia edit timestamps) and spatial knowledge from GeoNames[19]. YAGO 3 extends multilingual knowledge by utilizing Wikipedias inter-language links to cover additional values in many other languages. The latest version **YAGO 4** [200] no longer uses data from Wikipedia (in combination with Wordnet and GeoNames), but collects data from Wikidata [202] and forces it into a taxonomy based on schema.org[20] and Bioschemas [221]. SHACL constraints are used for classes to enforce a strict consistency. Manually defined mappings are applied from Wikidata to an initial set of 235 schema.org classes and 116 relations. The process iterates over each Wikidata item, filters low-coverage entities, and accepts entities and their types that are transitively connected to one of the initial classes via sub-class relations resulting in the final taxonomy of 10k classes (taxonomy enrichment). YAGO maintains fact provenance by using Wikidata's annotations for validity time or external references.

NELL [201]. The Never-Ending Language Learner is a system to incrementally construct a KG from text corpora and web pages. Per incremental execution (called iteration), it uses NLP-based knowledge extraction to determine entities, their types and relations between entities with the help of patterns. The central part of learning is the Coupled Pattern Learner (CPL), which memorizes patterns of the form *"X plays for Y"*. In each iteration, the system learns new patterns and simultaneously applies its previously learned patterns. Additionally, NELL generates metadata with newly learned patterns and rules. A user manually validates such newly learned patterns regularly (not necessarily for every iteration) before the pipeline uses them as future supplementary input data. While NELL generates neither RDF nor PGM data, the Nell2RDF [222] extension transforms its data to RDF and annotates extracted relations with provenance information about the used Wikipedia articles.

Artist-KG [204]. Gawriljuk et al. create a KG of artists from four different sources that are incrementally added one after the other. Per iteration (source), they first filter out artist entities and apply schema mapping with the Karma approach [111] to map entity properties. Then they perform entity resolution using artist name and birthdate similarities and utilize MinHash/LSH blocking to make this process scalable. They only apply the union of matching entities and leave entity fusion (value consolidation) to consuming applications. The final KG is in a custom graph format, serialized as JSON with with nested representations of artist entities.

AI-KG [206]. The Artificial Intelligence KG contains over 820K entities derived from over 333k research publications integrating and enriching data from the Microsoft Academic Graph [214], the Computer Science Ontology [223], and Wikidata. Its internal ontology builds on SKOS[21], PROV-O[22], and OWL. Fact provenance is supported by keeping identifiers to the original papers and information about the applied extraction tool. The RDF data of the KG is available via a publicly accessible Virtuoso triple store[23].

The pipeline has four components: 1) extractors, 2) entities handler, 3) relations handler, and 4) triple selector. The extractors use the NLP tools DyGIE++ [224], the CSO Classifier [225], and Stanford CoreNLP [118] to extract entities and relations from text for predefined entity types and relations. To determine possible entities and relations, the system operates an overlapping strategy between the CoreNLP tools like Open Information Extraction, its POS-Tagger, and the topics detected by the CSO Classifier. Semantically similar entities are clustered (via word embeddings and hierarchical clustering) and manually revised. The Triple Selector categorizes facts into valid and non-valid triples. For relations extracted based on patterns between entities derived by the POS-Tagger, an occurrence frequency threshold is used to determine the validity. By contrast, triples extracted by DyGie++ are trusted as this tool has achieved high precision in previous benchmarks.

While the authors planned periodic KG updates, the update process has not been described. The AI-KG was subsequently replaced by the CS-KG [226].

CovidGraph [207]. The open domain-specific CovidGraph integrates over 17 Covid-related sources on publications, authors, genes, proteins, and diseases from publication archives, databases, or open accessible APIs. The KG

[19] https://www.geonames.org/
[20] https://schema.org/
[21] https://www.w3.org/2004/02/skos/
[22] https://www.w3.org/TR/prov-o/
[23] http://scholkg.kmi.open.ac.uk/

is managed as a Neo4J database. For each data source, KG construction uses a modular process where the integration of every source dataset is handled by a separate Docker container dealing with source-specific data cleaning and mapping tasks. The final orchestration and order of integration is managed by a separate container[24]. Part of the data is extracted from biomedical paper abstracts via the biomedical language representation model BioBert [227]. The applied matching techniques use string similarities or usable global identifiers like symbols and ids, or for example the Reference Sequence Database [228] for genes. The KG contains provenance information about the originating paper, and nodes have modification timestamps. The project provides a schema graph image for exploration purposes but it remains unclear how the underlying ontology (schema) is maintained.

Due to the global impact of the Covid disease, similar projects emerged like the COVID literature KG [229] which uses other integration methods.

DRKG [208]. The Drug Repurposing KG integrates several biomedical sources (mostly open-access databases) with information about genes, proteins, diseases, and drugs. Its use cases are centered around drug redesign and re-purposing, e.g., utilizing newly discovered relations in the KG. Each data source is mapped to a triple structure using crowd-sourced tools like Bio2RDF[25] and rule-based mapping languages. Bio2RDF converts data to a crowd-sourced ontology, thereby providing some initial types and relationships. To resolve entities from different sources, the DRKG subsequently tries to map entities of the same type to a common ID space, e.g., the MESH-ID space for diseases. The final entity IDs contain the originating source thereby supporting backtracking the origin of facts in the KG. The DRKG project also applies link prediction by utilizing graph embeddings with the TransE model [189]. This was used to predict relations between drugs, genes and the three diseases SARS, MERS, and SARS-COV2 (COVID-19).

VisualSem [209] is a multilingual and multi-modal KG that interlinks images, their descriptions (glosses) and other attributes from Wikipedia articles, WordNet concepts, and ImageNet [230] images. The approach applies a combination of data retrieval, sampling, and cleaning from the ImageNet dataset. First an initial set of nodes is retrieved from ImageNet[26] for 1000 classes from the "ImageNet Large Scale Visual Recognition Challenge". Then, in an iterative process additional nodes are collected in the following way: 1) collect neighboring nodes of the current node pool 2) filter images (removing ones not meeting certain quality criteria 3) filter nodes (keep nodes with at least one image and two relation types) 4) update the node pool (accepting only top-k nodes). In more detail, during the first step relations from the ImageNet graph are mapped to 13 relationship names in the final graph via manually defined mappings. In the image cleaning step four filters where applied. The system checks for valid image files, removes duplicated images via SHA1 hashing, uses a ResNET-based binary classifier [231] to remove non-photographic images, and leverages OpenAI's CLIP to remove image that do not minimally match any of the node glosses. The iteration holds after reaching a node pool size of 90k. The project is accessible in a public git repository[27] and also contains pretrained models for vision and language research.

WorldKG [210] integrates the semi-structured data of OpenStreetMap (OSM)[28] into a geographic KG. The construction of WorldKG consists of two parts. The WorldKG ontology (initial KG) is constructed in the first part based on OSM's "Map" system and OSM wiki data. The Map system allows users to tag nodes, ways, or relations with geographic attributes encoded as key-value pairs, and the OSM wiki data describes these tags. In a scraping step, all tags that encode geographic class information are fetched from the site, then the obtained key-value pairs are used to infer a class hierarchy. For example, *school=building* resolves to *schoolBuilding* as a sub type of *building*. All classes from the initial ontology are then aligned (ontology matching) with Wikidata and DBpedia classes using a unsupervised machine learning approach [232]. The resulting class alignments are manually verified and cleaned. In the second part, the construction processes maps OSM data to the final KG structure performing three steps: 1) filter nodes with at least one tag, 2) filter keys and values based on the initial ontology, and 3) create RDF triples.

[24] https://git.connect.dzd-ev.de/dzdtools/motherlode
[25] https://github.com/bio2rdf/bio2rdf-scripts
[26] ImageNet is an image database organized according to the WordNet hierarchy.
[27] https://github.com/iacercalixto/visualsem
[28] https://www.openstreetmap.org

4.3. KG Frameworks & Strategies

DBpedia FlexiFusion [90] provides a workflow based on the DBpedia Databus platform to fuse Linked Open Data (LOD) datasets into a provenance-rich and uniform knowledge graph. Users have to register the individual datasets (downloadable dumps) as well as entity matches (link sets) across these datasets, either utilizing sets of `sameAs` relations of the LOD or by first generating such match links by a suitable entity resolution tool. Moreover, Non-RDF data has to be transformed to RDF by users. FlexiFusion generates a so-called PreFusion dump per dataset and uses the linked entities to form a cluster using the connected components algorithm. Matching entities of a cluster are then fused into a single entity for which the property values are selected, e.g. by taking the values from preferred sources. The fusion process distinguishes properties to have exactly one or multiple values depending on the property cardinality in the originating sources.

Two public KGs have been generated with FlexiFusion: 1) Global DBpedia, which integrates and fuses over 175 sources, including the 140 language-specific DBpedia extractions as well as more than 25 other LOD datasets. 2) an effort to create a Dutch National KG[29] integrating eight sources. For both projects, DBpedia (and its ontology) served as the initial KG and the existing links from the other datasets to DBpedia were used for the clustering and fusion steps.

dstlr [137] is a framework for scalable KG construction, relying on Apache Solr[30] as document store, an extraction and completion layer build on Apache Spark[31] and Neo4j as graph database for the resulting KG. The approach utilizes Stanford CoreNLP for all their knowledge extraction steps: Named-entity extraction, relation extraction and linking the extracted mentions to Wikidata. The framework keeps the provenance of those mentions w.r.t to the source documents. The resulting KG is enriched with facts from an external KG by manually defining mappings between CoreNLP relations and Wikidata properties and extracting corresponding facts from this external resource. The authors mention the possibility of utilizing Cypher queries to verify information in the KG against Wikidata information. While the authors note, that using Apache Solr as document store enables the ingestion of new batches of documents it is unclear how well the pipeline is capable of handling scenarios that can occur in an incremental paradigm, e.g., the deletion or updating of entities.

XI Pipeline [50]. This approach focuses on semi-automatically constructing KGs from unstructured or semi-structured documents such as publications and social network content. In the initial step, the system applies named-entity recognition on textual input data. Probabilistic models and microtask crowdsourcing are combined to accomplish entity linking with a quality outperforming models without a human-in-the-loop paradigm [233]. A type ranking step is performed to achieve fine-grained type associations of entities by leveraging the textual surrounding of entities as well as the current KG type hierarchy [234]. Relation Extraction utilizes distant supervision and an aggregated piecewise convolution network which is trained on existing relations of the KG.

XI is used for several KG projects. ScienceWise [235] annotates papers with a human-in-the-loop approach to build a research KG. ArmaTweet [236] enables the detection of events, such as natural disasters or terrorist activity, based on anomalies in extracted Twitter tweets. Guider [237] extracts a dependency graph from logs for dependency-driven analytics. The foundation of the KG is a manually developed ontology. Guiders metadata tracks multi-level granular provenance over facts, events, sources (posts, users, locations). In some use cases, the XI functionality has been extended, e.g. for metadata support in Guider.

AutoKnow [196] is a closed-source approach for creating a product KG in the domain of retail products at Amazon. The system processes existing product catalogs and consumer shopping behavior logs leveraging several machine learning approaches and distant supervision for training. Its architecture consists of a so-called ontology suite and a data suite. The ontology suite performs taxonomy enrichment (extraction, attachment) as well as relation discovery. In the taxonomy enrichment step, new types are extracted from the input product catalogs and customer queries [238]. A GNN approach is applied to place new types into the existing ontology. Relations are extracted using classification models for attribute applicability and a regression model for attribute importance on product profiles and the user's search, review, or Q&A data.

[29] https://github.com/dbpedia/dnkg-pilot
[30] https://solr.apache.org/
[31] https://spark.apache.org/index.html

The data suite performs data imputation (knowledge extraction), data cleaning, and synonym finding. Data imputation extracts attribute-value pairs from the product data using a taxonomy-aware tagging approach. It leverages CRF combined with multi-task learning with a shared BiLSTM to simultaneously train sequence tagging and product type categorization. The data cleaning phase checks extracted attribute-value pairs for correctness based on a transformer-based neural net model. Synonym finding is based on a supervised approach using a combination of collaborative filtering and a simple logistic regression model.

Most of the training and validation data are derived from the product catalog or customer behavior logs applying distant supervision and, in some cases, utilizing crowdsourcing with Amazon MTurk. In an experimental execution, AutoKnow backed the construction of a product graph at Amazon with more than 30 million entities and 1 billion relations assigned to 19K entity types and 1K relation types.

HKGB [211]. The *Health Knowledge Graph Builder* is a platform to semi-automatically construct clinical KGs with heavy human-in-the-loop (HL) involvement. As input, the system consumes Electronic Medical Records (EMR) consisting of structured and unstructured parts. It semi-automatically processes the data in combination with the clinicians' inputs and produces graph data in OWL and RDF. The human interaction involves: 1) new concept/relation inspection (approving recommended concepts or relations), 2) adding medical synonym entities based on instances, 3) the annotation of unstructured data based on instances and relations of the current KG, and 4) the definition of mapping rules from EMR to RDF and the extraction of concepts, entities, and relations. The HL inspection mainly contributes to the ontology construction and all other steps to every aspect of the KG (instances and ontology). Annotations are accepted if there is high agreement across annotators above some confidence threshold (0.81). Disease-specific information ingestion divides into two phases: 1) building the Concept KG (ontology), and 2) building the Instance KG. The authors describe an incremental process to add other diseases afterward using a similar strategy (similar to the initial construction). In addition to the construction tools, the HKGB platform provides three graph tools for data discovery, extraction, and link prediction to support domain-related applications.

The HKGB was utilized to develop the HuadingKG, which consists of about 85 million entities and 265 million relations, initially built with information about cardiovascular diseases and later enriched with information from the Knee Osteoarthritis domain.

SLOGERT [212]. *Semantic LOG ExtRaction Templating* is a framework for automated KG construction from log data. The resulting KGs are utilized in security-related applications to detect upcoming threats and vulnerabilities. It heavily uses known tools, e.g., LOGPAI for log file pattern parsing, Stanford NLP for entity recognition, and the OTTR Engine (LUTRA) with Apache Jena to manage RDF data. As the KGs foundation, the internal ontology extends a previous vocabulary [239] with mappings to the Common Event Expression[32] taxonomy. The implementation is publicly available in a git repository[33] and the log ontology is shared online[34].

The workflow runs in two phases. The first phase extracts data and parameters from log files and generates RDF templates conforming to the KG ontology. For named entities determined by Stanford CoreNLP type and properties are defined from a log vocabulary. In the second phase, the extracted template information is used to convert each log file to a graph, and then the log file graphs are integrated into the final KG. This is done by utilizing appropriate identifiers to connect local context information (e.g., network architecture or organizational structures) to key concepts and identifiers of the computer log domain (IP, MAC, URL) and external sources such as vulnerability databases or service inventories. If external knowledge does not align, an entity resolution step is required; the authors mention the SILK framework [240] for this task.

SAGA [47]. This closed-source toolset supports multi-source data integration for both batch-like incremental KG construction and continuous KG updates. The internal data model extends standard RDF to capture one-hop relationships among entities, provenance (source), and trustworthiness of values. The system supports source change detection and delta computation using their last snapshots. Based on detected changes, SAGA executes parallel batch jobs to integrate an updated or new source into its target graph. SAGA's ingestion component requires mappings from new data to the internal KG ontology. This step only requires predicate mappings, as the subject and object fields can remain in their original namespace and are linked later in the process. The required mappings are mostly

[32] https://cee.mitre.org/language/
[33] https://github.com/sepses/slogert
[34] https://w3id.org/sepses/ns/log

manually defined and stored as supplementary configuration files. Additionally, data can be reprocessed with the HoloClean tool [81] for data repair. SAGA is able to detect and disambiguate entities from text and (semi-)structured sources. To make the deduplication step scalable it groups entities by type and performs simple blocking to further partition the data into smaller buckets. A matching model computes similarity scores and machine-learning- or rule-based methods are applicable to determine likely matches. Correlation clustering [241] is then utilized to determine matching entities. The system tracks same-as links to original source entities to support debugging. For entity fusion, (conflicting) entity attribute values are harmonized based on truth discovery methods and source reliability to create consistent entities.

In addition to the stable KG (updated in batches), the system can maintain a *Live Graph*, which continuously integrates streaming data and whose entities reference the stable entities of the batch-based KG. For scalability and near-real-time query performance, the live graph uses an inverted index and a key-value store. SAGA supports live graph curation by using a human-in-the-loop approach. The authors mention that SAGA powers question answering, entity summarization, and text annotation (NER) services.

5. Discussion & Open Challenges

Our study of existing solutions for KG construction showed that there are many different approaches not only for building specific KGs but also in the current toolsets. This underlines the inherent complexity of the problem and the dependency on different criteria such as the major kinds of input data and the intended use cases. The requirements we posed in Section 2.3 are also not yet met to a larger degree indicating the need for more research and development efforts. This is also because there are inherent tradeoffs between the goals of high data quality, scalability and automation [9] that ask for compromise solutions. So while it is possible to have a large degree of automation for the individual construction tasks, human interaction generally tends to improve the quality significantly. On the other hand, such human interaction can become a limiting factor towards scalability to many sources and high data volume.

In the following, we discuss open challenges and areas for future work on KG construction that we see. The focus here is on broader issues rather than specific limitations in individual steps.

Incremental KG Construction We observed that most construction pipelines for specific KGs and in toolsets do not yet support incremental KG updates but are limited to a batch-like re-creation of the entire KG. As already discussed in the requirement section 2.3 this approach has significant limitations and prevents scalability to many data sources and high data volume. We therefore need better support for incremental KG updates especially in toolsets. Such a capability has to provide solutions to a variety of issues. As already discussed in Sec. 3.1.2, it has to be detected *if* there are changes in the input data and if so to determine *what* has changed. The changes to be dealt with are not limited to the addition of new information but deletions and updates in the sources have to be propagated to the KG as well. Changes that impact the underlying ontology or the pipeline's configuration also have to be managed and may require manual interaction/confirmation. Support for a streaming-like propagation of source changes should also increase so that KGs can provide the most recent information.

Lack of open tools As we have seen in Table 4 most KG construction toolsets, especially the more advanced ones, are closed source and can thus not be used by others for creating a KG or for evaluating their functionality. Hence there is a strong need for more open-source toolsets to help improve the development of KGs and to advance the state-of-the-art in KG construction. Researchers and developers providing such an implementation and associated publications could achieve a high impact [242].

Improved Extensibility & Modularisation, Ease of Use A toolset for KG construction should be able to define and execute different pipelines depending on the data sources to be integrated and specific requirements, e.g., for incremental updates. Hence, an extensible and modular KG construction approach should be provided with alternate implementations for the different KG construction tasks to choose from. This can be facilitated by making use of existing implementations as has been done already for NLP tasks (e.g., Stanford CoreNLP) but not yet for other tasks such as entity resolution. From the projects compared in Section 4 only a few addressed this problem so that more solutions are needed.

The definition of a KG construction pipeline should be relatively easy, supported by a user-friendly GUI and with a low effort for configuring the pipeline and its individual tasks. The configuration can be simplified by providing default settings for most parameters or even automatic approaches based on machine learning [243]. On the other hand, a manual configuration should also be possible to achieve customizability and support for special cases (e.g., a new entity type or input format). Extensibility and modularisation of a tool should not lead to a higher configuration effort for users.

Data and metadata management. Good data and metadata management is vital in an open and incremental KG process. Only a few solutions even mention an underlying management architecture supporting the construction processes. Having uniform access or interfaces to data and relevant metadata can drastically improve the quality of the former [83] and increases the workflow's replicability and possibilities for debugging. A dedicated metadata repository can store used mappings, schemata, and quality reports, improving the transparency of the entire pipeline process.

Metadata support is limited in current solutions and only some pipeline approaches acknowledge the importance of provenance tracking and debugging possibilities. We found that the term provenance is rather vaguely used, mostly in the meaning of tracking the source of facts and the processes involved in their generation. Only few approaches such as SAGA [47] also try to maintain the trustworthiness of facts. Metadata such as fact-level provenance should be used more to support construction tasks, such as for data fusion to determine final entity values. In general there is a need for maintaining more metadata, especially temporal information, that is also essential for studying the evolution of KG information. Support for developing temporal KGs maintaining historical and current data, compared to the common sequences of static KG snapshot versions, is also a promising direction.

Data Quality One of the main goals of KG construction is to achieve and maintain a high quality KG. The difficulty of this task grows with rising number and heterogeneity of data sources, especially if one relies on automatic data acquisition and data integration. High-quality sources can provide a clean type hierarchy and can serve as training data to alleviate some data-quality issues, that would be more difficult to address by treating low-quality sources in isolation [9]. Lower quality data sources often contain a high degree of long tail entities (which is the reason these data source are valuable). Nevertheless, corroborating the information from these sources with evidence from higher-quality sources remains difficult and can reduce data quality. For example, the automatic fusion of conflicting entity values can easily introduce wrong information into a KG and even a restricted degree of human intervention is problematic on a large scale [114, 244].

To achieve the best possible data quality, data cleaning should be part of all major steps in the construction pipeline so that the degree of dirty or wrong information that is entering the KG is limited. Moreover, the identification and repair of errors should be a continuous task especially in large KG projects [245]. To better address these problems, more comprehensive data quality measure and repair strategies are needed that minimize human intervention to retain high scalability for KG construction.

Evaluation The evaluation of complete KG construction pipelines is an open but important problem to measure the performance and quality of current approaches and to improve on them. So far there are benchmarks for individual tasks such as knowledge extraction [246–248], ontology matching [249], entity resolution [150, 151, 250, 251] and KG completion [252–254]. While these benchmarks in some cases still leave gaps, e.g, regarding scalability [145] or domain diversity [255], they are already quite complex and indicate the high difficulty of defining a benchmark for entire KG construction pipelines.

A benchmark could be based on similar settings than for the creation of specific KGs discussed in Section 4 aiming at the initial construction and incremental update of either a domain-specific or cross-domain KG from a defined set of data sources of different kinds. The KG ontology and the KG data model (RDF or property graph) could be predefined to facilitate the evaluation of the resulting KG. The size of the resulting KG should be relatively high and the construction should be challenging with the need of knowledge extraction, entity linking/resolution and entity fusion. Determining the quality of the constructed KG is difficult as it would ideally be based on a near-perfect result (gold standard) for the initial KG and for its updated version(s). For all entity and relation types in the given KG ontology, it has then to be determined to what degree they could correctly be populated compared to the gold standard which requires extension to known metrics such as precision and recall. Further evaluation criteria include the runtimes for the initial KG construction and for the incremental updates and perhaps the manual effort to set up

the construction pipelines. Ideally, an evaluation platform could be established - similar to other areas [256] - for a comparative evaluation of different pipelines with different implementations of the individual construction steps.

The whole is more than the sum of it's parts. While the individual parts of KG construction pipelines are well-established research problems with sometimes decades of previous research, the complex interaction of the pipeline tasks is not well researched yet. For example, the disambiguation strategies of the knowledge extraction task, especially entity linking, are very similar to entity resolution. The use of background knowledge and various inter-dependencies between different information is commonly summarized as *holistic entity linking*. This approach has seen some research attention and a survey with future research directions was published by Oliveira et. al. in a 2021 paper [257]. While such approaches go in the right direction, our pipeline scenario would invite an even more holistic case, where named-entity linking and entity resolution approaches aid each other to boost their performance. Furthermore, data cleaning can independently be done in several tasks but it would be beneficial to have a coordinated approach to avoid duplicate efforts.

6. Conclusion

This work presented the current state of knowledge graph construction, giving an overview of the requirements and defining this area's central concepts and tasks. We gave a synopsis of techniques used to address individual steps of such a pipeline with a perspective on how well the state-of-the-art solutions for these specific tasks can be integrated into an incremental KG construction approach. We comparatively analysed a selection of current KG-specific pipelines and toolsets for KG construction, based on a list of criteria derived from our initial requirements set. We found vast differences across these pipelines concerning the number and structure of the input data, applied construction methods, ontology management, the ability to continuously integrate new information and the tracking of provenance throughout the pipeline. The open KG-specific approaches are currently rather limited in their scalability to many sources, support for incremental updates and in several steps regarding metadata, ontology management, entity resolution / fusion, and quality assurance. The considered toolsets are generally better in terms of their functionality but they are mostly closed-source and thus not usable for new KG projects or research investigations.

We identified various challenges that need to be addressed for improved incremental KG construction. These problems range from engineering questions like the need for a flexible software architecture, over numerous task-specific problems and the support for incremental construction, to hurdles that must be addressed collectively by the research community, like the development of open-source and modular toolsets for KG construction and on benchmarking and evaluation processes. Concerning the exploitation of new data sources, integrating more multimodal data is of great potential but also requires more research to achieve effective solutions.

Addressing the derived challenges promises significant advances for future KG construction pipelines and much reduced effort for creating and maintaining high-quality KGs.

Acknowledgements. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI

References

[1] 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, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, J.S. Culpepper, A. Moffat, P.N. Bennett and K. Lerman, eds, ACM, 2019, pp. 105–113. doi:10.1145/3289600.3290956.

[2] X. Wang, X. He, Y. Cao, M. Liu and T. Chua, KGAT: Knowledge Graph Attention Network for Recommendation, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi and G. Karypis, eds, ACM, 2019, pp. 950–958. doi:10.1145/3292500.3330989.

[3] S.K. Mohamed, V. Nováček and A. Nounu, Discovering protein drug targets using knowledge graph embeddings, *Bioinformatics* **36**(2) (2019), 603–610. doi:10.1093/bioinformatics/btz600.
[4] H. Oberkampf, S. Zillner and B. Bauer, Interpreting Patient Data using Medical Background Knowledge, in: *Proceedings of the 3rd International Conference on Biomedical Ontology (ICBO 2012), KR-MED Series, Graz, Austria, July 21-25, 2012*, R. Cornet and R. Stevens, eds, CEUR Workshop Proceedings, Vol. 897, CEUR-WS.org, 2012. http://ceur-ws.org/Vol-897/session2-paper06.pdf.
[5] D. Sonntag, V. Tresp, S. Zillner, A. Cavallaro, M. Hammon, A. Reis, P.A. Fasching, M. Sedlmayr, T. Ganslandt, H. Prokosch, K. Budde, D. Schmidt, C. Hinrichs, T. Wittenberg, P. Daumke and P.G. Oppelt, The Clinical Data Intelligence Project - A smart data initiative, *Inform. Spektrum* **39**(4) (2016), 290–300. doi:10.1007/s00287-015-0913-x.
[6] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proceedings of the IEEE* **104**(1) (2015), 11–33.
[7] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J.E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J.F. Sequeda, S. Staab and A. Zimmermann, Knowledge Graphs (2021). ISBN 9781636392363. doi:10.2200/S01125ED1V01Y202109DSK022. https://kgbook.org/.
[8] S. Ji, S. Pan, E. Cambria, P. Marttinen and S.Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, *IEEE Transactions on Neural Networks and Learning Systems* **33**(2) (2021), 494–514.
[9] G. Weikum, L. Dong, S. Razniewski and F.M. Suchanek, Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases, *Found. Trends Databases* **10** (2021), 108–490.
[10] X. Zhu, Z. Li, X. Wang, X. Jiang, P. Sun, X. Wang, Y. Xiao and N.J. Yuan, Multi-Modal Knowledge Graph Construction and Application: A Survey, *ArXiv* **abs/2202.05786** (2022).
[11] V. Ryen, A. Soylu and D. Roman, Building Semantic Knowledge Graphs from (Semi-) Structured Data: A Review, *Future Internet* **14**(5) (2022), 129.
[12] X. Ma, Knowledge graph construction and application in geosciences: A review, *Comput. Geosci.* **161** (2021), 105082.
[13] G. Tamašauskaitė and P. Groth, Defining a Knowledge Graph Development Process Through a Systematic Review, *ACM Transactions on Software Engineering and Methodology* (2022).
[14] E.W. Schneider, Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis. (1973).
[15] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic web* **8**(3) (2017), 489–508.
[16] L. Ehrlinger and W. Wöß, Towards a Definition of Knowledge Graphs., in: *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.
[17] M. Lissandrini, D. Mottin, K. Hose and T.B. Pedersen, Knowledge Graph Exploration Systems: are we lost?, in: *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*, www.cidrdb.org, 2022. https://www.cidrdb.org/cidr2022/papers/p40-lissandrini.pdf.
[18] A. Hogan, D. Brickley, C. Gutierrez, A. Polleres and A. Zimmerman, (Re)Defining Knowledge Graphs, in: *Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371)*, Vol. 8, 2018, pp. 74–79. doi:10.4230/DagRep.8.9.29.
[19] C. Feilmayr and W. Wöß, An analysis of ontologies and their success factors for application to business, *Data & Knowledge Engineering* **101** (2016), 1–23.
[20] S. Sakr, A. Bonifati, H. Voigt, A. Iosup, K. Ammar, R. Angles, W.G. Aref, M. Arenas, M. Besta, P.A. Boncz, K. Daudjee, E.D. Valle, S. Dumbrava, O. Hartig, B. Haslhofer, T. Hegeman, J. Hidders, K. Hose, A. Iamnitchi, V. Kalavri, H. Kapp, W. Martens, M.T. Özsu, E. Peukert, S. Plantikow, M. Ragab, M. Ripeanu, S. Salihoglu, C. Schulz, P. Selmer, J.F. Sequeda, J. Shinavier, G. Szárnyas, R. Tommasini, A. Tumeo, A. Uta, A.L. Varbanescu, H. Wu, N. Yakovets, D. Yan and E. Yoneki, The future is big graphs: a community view on graph processing systems, *Commun. ACM* **64**(9) (2021), 62–71. doi:10.1145/3434642.
[21] O. Lassila, Resource description framework (RDF) model and syntax specification, W3C recommendation, *http://www. w3. org/TR/PR-rdf-syntax* (1999).
[22] H. Knublauch and D. Kontokostas, Shapes constraint language (SHACL), *W3C Candidate Recommendation* **11**(8) (2017).
[23] E. Prud'hommeaux, J.E.L. Gayo and H.R. Solbrig, Shape expressions: an RDF validation and transformation language, in: *Joint Conference on Lexical and Computational Semantics*, 2014.
[24] J. Frey, K. Müller, S. Hellmann, E. Rahm and M.-E. Vidal, Evaluation of metadata representations in RDF stores, *Semantic Web* **10**(2) (2019), 205–229.
[25] L.F. Sikos and D. Philp, Provenance-aware knowledge representation: A survey of data models and contextualized knowledge graphs, *Data Science and Engineering* **5**(3) (2020), 293–316.
[26] F. Zhang, Z. Li, D. Peng and J. Cheng, RDF for temporal data management – a survey, *Earth Science Informatics* **14** (2021), 563–599.
[27] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C.N. Ngomo and H. Jabeen, Distributed Semantic Analytics Using the SANSA Stack, in: *International Workshop on the Semantic Web*, 2017.
[28] R. Angles, The Property Graph Database Model, in: *AMW*, 2018.
[29] H. Lbath, A. Bonifati and R. Harmer, Schema inference for property graphs, in: *EDBT 2021-24th International Conference on Extending Database Technology*, 2021, pp. 499–504.
[30] N. Inc., Neo4j Graph Database, 2022. https://neo4j.com/.
[31] T.L. Foundation, JanusGraph: an open source, distributed graph database, 2022. https://janusgraph.org.
[32] I. TigerGraph, TigerGraph graph database, 2022. https://www.tigergraph.com.

[33] S. Hong, S. Depner, T. Manhardt, J. Van Der Lugt, M. Verstraaten and H. Chafi, PGX.D: A Fast Distributed Graph Processing Engine, in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, Association for Computing Machinery, New York, NY, USA, 2015. ISBN 9781450337236. doi:10.1145/2807591.2807620.
[34] C. Rost, K. Gómez, M. Täschner, P. Fritzsche, L. Schons, L. Christ, T. Adameit, M. Junghanns and E. Rahm, Distributed temporal graph analytics with GRADOOP, *VLDB J.* **31**(2) (2022), 375–401. doi:10.1007/s00778-021-00667-4.
[35] P.T. Wood, Query languages for graph databases, *SIGMOD Rec.* **41**(1) (2012), 50–60. doi:10.1145/2206869.2206879.
[36] R. Angles, M. Arenas, P. Barceló, P.A. Boncz, G.H.L. Fletcher, C. Gutierrez, T. Lindaaker, M. Paradies, S. Plantikow, J.F. Sequeda, O. van Rest and H. Voigt, G-CORE: A Core for Future Graph Query Languages, in: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, G. Das, C.M. Jermaine and P.A. Bernstein, eds, ACM, 2018, pp. 1421–1432. doi:10.1145/3183713.3190654.
[37] M.A. Rodriguez, The Gremlin graph traversal machine and language (invited talk), in: *Proceedings of the 15th Symposium on Database Programming Languages*, ACM, 2015. doi:10.1145/2815072.2815073. https://doi.org/10.1145%2F2815072.2815073.
[38] O. van Rest, S. Hong, J. Kim, X. Meng and H. Chafi, PGQL: a property graph query language, in: *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*, P.A. Boncz and J. Larriba-Pey, eds, ACM, 2016, p. 7. doi:10.1145/2960414.2960421.
[39] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer and A. Taylor, Cypher: An Evolving Query Language for Property Graphs, in: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, G. Das, C.M. Jermaine and P.A. Bernstein, eds, ACM, 2018, pp. 1433–1445. doi:10.1145/3183713.3190657.
[40] A. Deutsch, N. Francis, A. Green, K. Hare, B. Li, L. Libkin, T. Lindaaker, V. Marsault, W. Martens, J. Michels, F. Murlak, S. Plantikow, P. Selmer, O. van Rest, H. Voigt, D. Vrgoc, M. Wu and F. Zemke, Graph Pattern Matching in GQL and SQL/PGQ, in: *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Z. Ives, A. Bonifati and A.E. Abbadi, eds, ACM, 2022, pp. 2246–2258. doi:10.1145/3514221.3526057.
[41] H. Chiba, R. Yamanaka and S. Matsumoto, Property Graph Exchange Format, *ArXiv preprint* **abs/1907.03936** (2019). https://arxiv.org/abs/1907.03936.
[42] D. Tomaszuk, R. Angles, L. Szeremeta, K. Litman and D. Cisterna, Serialization for Property Graphs, in: *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis - 15th International Conference, BDAS 2019, Ustroń, Poland, May 28-31, 2019, Proceedings*, S. Kozielski, D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek and D. Kostrzewa, eds, Communications in Computer and Information Science, Vol. 1018, Springer, 2019, pp. 57–69. doi:10.1007/978-3-030-19093-4_5.
[43] R. Angles, A. Bonifati, S. Dumbrava, G. Fletcher, K. Hare, J. Hidders, V.E. Lee, B. Li, L. Libkin, W. Martens, F. Murlak, J. Perryman, O. Savkovic, M. Schmidt, J. Sequeda and D. Tomaszuk, PG-Keys: Keys for Property Graphs, *Proceedings of the 2021 International Conference on Management of Data* (2021).
[44] C. Rost, P. Fritzsche, L. Schons, M. Zimmer, D. Gawlick and E. Rahm, Bitemporal Property Graphs to Organize Evolving Systems, *arXiv preprint arXiv:2111.13499* (2021).
[45] M. Besta, M. Fischer, V. Kalavri, M. Kapralov and T. Hoefler, Practice of Streaming and Dynamic Graphs: Concepts, Models, Systems, and Parallelism, *CoRR* **abs/1912.12740** (2019). http://arxiv.org/abs/1912.12740.
[46] O. Lassila, M. Schmidt, O. Hartig, B. Bebee, D. Bechberger and W. Broekema, The OneGraph Vision: Challenges of Breaking the Graph Model Lock-In, *Semantic Web* (2022).
[47] I.F. Ilyas, T. Rekatsinas, V. Konda, J. Pound, X. Qi and M. Soliman, Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale, *ArXiv preprint* **abs/2204.07309** (2022). https://arxiv.org/abs/2204.07309.
[48] O. Hartig, Reconciliation of RDF* and Property Graphs, *CoRR* **abs/1409.3288** (2014). http://arxiv.org/abs/1409.3288.
[49] G. Abuoda, D. Dell'Aglio, A.K. Keen and K. Hose, Transforming RDF-star to Property Graphs: A Preliminary Analysis of Transformation Approaches - extended version, *ArXiv* **abs/2210.05781** (2022).
[50] P. Cudré-Mauroux, Leveraging knowledge graphs for big data integration: the XI pipeline, *Semantic Web* **11**(1) (2020), 13–17.
[51] S.E. Madnick, R.Y. Wang, Y.W. Lee and H. Zhu, Overview and Framework for Data and Information Quality Research, *ACM J. Data Inf. Qual.* **1**(1) (2009), 2:1–2:22. doi:10.1145/1515693.1516680.
[52] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann and S. Auer, Quality assessment for linked data: A survey, *Semantic Web* **7**(1) (2016), 63–93.
[53] X.L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun and W. Zhang, Knowledge-based trust: Estimating the trustworthiness of web sources, *ArXiv preprint* **abs/1502.03519** (2015). https://arxiv.org/abs/1502.03519.
[54] Y. Amsterdamer and M. Cohen, Automated Selection of Multiple Datasets for Extension by Integration, in: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 27–36.
[55] B. Fetahu, S. Dietze, B. Pereira Nunes, M. Antonio Casanova, D. Taibi and W. Nejdl, A scalable approach for efficiently generating structured dataset topic profiles, in: *European Semantic Web Conference*, Springer, 2014, pp. 519–534.
[56] D.M. Blei and J.D. Lafferty, A correlated topic model of science, *The annals of applied statistics* **1**(1) (2007), 17–35.
[57] M. Nentwig and E. Rahm, Incremental clustering on linked data, in: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2018, pp. 531–538.
[58] A. Saeedi, E. Peukert and E. Rahm, Incremental Multi-source Entity Resolution for Knowledge Graph Completion, in: *European Semantic Web Conference*, Springer, 2020, pp. 393–408.
[59] S. Hertling and H. Paulheim, Order Matters: Matching Multiple Knowledge Graphs, *ArXiv preprint* **abs/2111.02239** (2021). https://arxiv.org/abs/2111.02239.

[60] M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, Ö.L. Özçep and R. Rosati, Optique: Zooming in on Big Data, *Computer* **48**(3) (2015), 60–67. doi:10.1109/MC.2015.82.
[61] C. Civili, M. Console, G.D. Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli and D.F. Savo, MASTRO STUDIO: Managing Ontology-Based Data Access applications, *Proc. VLDB Endow.* **6**(12) (2013), 1314–1317. doi:10.14778/2536274.2536304. http://www.vldb.org/pvldb/vol6/p1314-poggi.pdf.
[62] M.N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer and J. Lehmann, Squerall: Virtual Ontology-Based Access to Heterogeneous and Large Data Sources, in: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I.F. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Lecture Notes in Computer Science, Vol. 11779, Springer, 2019, pp. 229–245. doi:10.1007/978-3-030-30796-7_15.
[63] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom and D.C. Sturman, An efficient multicast protocol for content-based publish-subscribe systems, in: *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, IEEE, 1999, pp. 262–272.
[64] M. Völkel and T. Groza, SemVersion: An RDF-based Ontology Versioning System, in: *Proceedings of IADIS International Conference on WWW/Internet*, IADIS, Murcia, Spain, 2006, IADIS. http://www.xam.de/2006/10-SemVersion-ICIW2006.pdf.
[65] D.-H. Im, S.-W. Lee and H.-J. Kim, A Version Management Framework for RDF Triple Stores., *International Journal of Software Engineering and Knowledge Engineering* **22**(1) (2012), 85–106. http://dblp.uni-trier.de/db/journals/ijseke/ijseke22.html#ImLK12.
[66] M. Graube, S. Hensel and L. Urbas, R43ples: Revisions for Triples - An Approach for Version Control in the Semantic Web, in: *LDQ@SEMANTiCS*, CEUR Workshop Proceedings, Vol. 1215, CEUR-WS.org, 2014.
[67] M.V. Sande, P. Colpaert, R. Verborgh, S. Coppens, E. Mannens and R.V. de Walle, R&Wbase: git for triples, in: *LDOW*, CEUR Workshop Proceedings, Vol. 996, CEUR-WS.org, 2013.
[68] D. Im, S. Lee and H. Kim, A Version Management Framework for RDF Triple Stores, *Int. J. Softw. Eng. Knowl. Eng.* **22**(1) (2012), 85–106.
[69] T. Neumann and G. Weikum, X-RDF-3X: Fast Querying, High Update Rates, and Consistency for RDF Databases, *Proc. VLDB Endow.* **3**(1–2) (2010), 256–263–. doi:10.14778/1920841.1920877.
[70] K. Stefanidis, I. Chrysakis and G. Flouris, On Designing Archiving Policies for Evolving RDF Datasets on the Web, in: *ER*, Lecture Notes in Computer Science, Vol. 8824, Springer, 2014, pp. 43–56.
[71] A. Dimou, R2RML and RML Comparison for RDF Generation, their Rules Validation and Inconsistency Resolution, *arXiv preprint arXiv:2005.06293* (2020).
[72] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, Rdf mapping language (rml), *Specification proposal draft* (2014).
[73] N. Jain, G. Liao and T.L. Willke, Graphbuilder: scalable graph ETL framework, in: *First Int. Workshop on Graph data management experiences and systems*, 2013.
[74] M. Kricke, E. Peukert and E. Rahm, Graph data transformations in Gradoop, *BTW 2019* (2019).
[75] R. Angles, H. Thakkar and D. Tomaszuk, Mapping RDF databases to property graph databases, *IEEE Access* **8** (2020), 86091–86110.
[76] E. Rahm and H.H. Do, Data cleaning: Problems and current approaches, *IEEE Data Eng. Bull.* **23**(4) (2000), 3–13.
[77] Z. Abedjan, X. Chu, D. Deng, R.C. Fernandez, I.F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker and N. Tang, Detecting Data Errors: Where are we and what needs to be done?, *Proc. VLDB Endow.* **9**(12) (2016), 993–1004. doi:10.14778/2994509.2994518. http://www.vldb.org/pvldb/vol9/p993-abedjan.pdf.
[78] I.F. Ilyas and X. Chu, *Data cleaning*, Morgan & Claypool, 2019.
[79] M. Fiorelli and A. Stellato, Lifting Tabular Data to RDF: A Survey, in: *Metadata and Semantic Research*, E. Garoufallou and M.-A. Ovalle-Perandones, eds, Springer International Publishing, Cham, 2021, pp. 85–96. ISBN 978-3-030-71903-6.
[80] Z. Abedjan, L. Golab, F. Naumann and T. Papenbrock, Data profiling, *Synthesis Lectures on Data Management* **10**(4) (2018), 1–154.
[81] T. Rekatsinas, X. Chu, I.F. Ilyas and C. Ré, HoloClean: Holistic Data Repairs with Probabilistic Inference, *Proc. VLDB Endow.* **10**(11) (2017), 1190–1201. doi:10.14778/3137628.3137631. http://www.vldb.org/pvldb/vol10/p1190-rekatsinas.pdf.
[82] F. Neutatz, B. Chen, Z. Abedjan and E. Wu, From Cleaning before ML to Cleaning for ML., *IEEE Data Eng. Bull.* **44**(1) (2021), 24–41.
[83] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3**(1) (2016), 1–9.
[84] M. Kricke, M. Grimmer and M. Schmeißer, Preserving Recomputability of Results from Big Data Transformation Workflows, *Datenbank-Spektrum* **17**(3) (2017), 245–253.
[85] J. Greenberg, Understanding metadata and metadata schemes, *Cataloging & classification quarterly* **40**(3–4) (2005), 17–36.
[86] C.B. Neto, D. Kontokostas, A. Kirschenbaum, G.C. Publio, D. Esteves and S. Hellmann, IDOL: Comprehensive & complete LOD insights, in: *Proceedings of the 13th International Conference on Semantic Systems*, 2017, pp. 49–56.
[87] E. Duval, W. Hodgins, S. Sutton and S.L. Weibel, Metadata principles and practicalities, *D-lib Magazine* **8**(4) (2002), 1–10.
[88] D. Kadioglu, B. Breil, C. Knell, M. Lablans, S. Mate, D. Schlue, H. Serve, H. Storf, F. Ückert, T.O. Wagner et al., Samply. MDR-A Metadata Repository and Its Application in Various Research Networks., in: *GMDS*, 2018, pp. 50–54.
[89] J. Frey, F. Götz, M. Hofer and S. Hellmann, Managing and Compiling Data Dependencies for Semantic Applications Using Databus Client, in: *Research Conference on Metadata and Semantics Research*, Springer, 2022, pp. 114–125.
[90] J. Frey, M. Hofer, D. Obraczka, J. Lehmann and S. Hellmann, DBpedia FlexiFusion the best of Wikipedia> Wikidata> your data, in: *International Semantic Web Conference*, Springer, 2019, pp. 96–112.
[91] N.F. Noy, D.L. McGuinness et al., Ontology development 101: A guide to creating your first ontology, Stanford knowledge systems laboratory technical report KSL-01-05 and . . . , 2001.

[92] F.N. Al-Aswadi, H.Y. Chan and K.H. Gan, Automatic ontology construction from text: a review from shallow to deep learning trend, *Artificial Intelligence Review* **53**(6) (2020), 3901–3928.
[93] A. Browarnik and O. Maimon, Ontology learning from text: why the ontology learning layer cake is not viable, *International Journal of Signs and Semiotic Systems (IJSSS)* **4**(2) (2015), 1–14.
[94] W. Wong, W. Liu and M. Bennamoun, Ontology learning from text: A look back and into the future, *ACM Computing Surveys (CSUR)* **44**(4) (2012), 1–36.
[95] C. Ma and B. Molnár, Ontology learning from relational database: Opportunities for semantic information integration, *Vietnam Journal of Computer Science* **9**(01) (2022), 31–57.
[96] R. De Virgilio, A. Maccioni and R. Torlone, R2G: a Tool for Migrating Relations to Graphs., in: *EDBT*, Vol. 2014, 2014, pp. 640–643.
[97] A. Petermann, M. Junghanns, R. Müller and E. Rahm, BIIIG: enabling business intelligence with integrated instance graphs, in: *2014 IEEE 30th International Conference on Data Engineering Workshops*, IEEE, 2014, pp. 4–11.
[98] D. Obraczka, A. Saeedi and E. Rahm, Knowledge Graph Completion with FAMER (DI2KG Challenge Winner), in: *Proceedings of the 1st International Workshop on Challenges and Experiences from Data Integration to Knowledge Graphs co-located with the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019), Anchorage, Alaska, August 5, 2019*, D. Firmani, V. Crescenzi, A.D. Angelis, X.L. Dong, M. Mazzei, P. Merialdo and D. Srivastava, eds, CEUR Workshop Proceedings, Vol. 2512, CEUR-WS.org, 2019. http://ceur-ws.org/Vol-2512/paper1.pdf.
[99] F.M. Suchanek, S. Abiteboul and P. Senellart, PARIS: Probabilistic Alignment of Relations, Instances, and Schema, *Proc. VLDB Endow.* **5**(3) (2011), 157–168. doi:10.14778/2078331.2078332. http://www.vldb.org/pvldb/vol5/p157_fabianmsuchanek_vldb2012.pdf.
[100] E. Rahm and P.A. Bernstein, A survey of approaches to automatic schema matching, *the VLDB Journal* **10**(4) (2001), 334–350.
[101] J. Euzenat, P. Shvaiko et al., *Ontology matching*, Vol. 18, Springer, 2007.
[102] Z. Bellahsene, A. Bonifati and E. Rahm, *Schema matching and mapping*, Springer, 2011.
[103] E. Rahm, Towards Large-Scale Schema and Ontology Matching, *Schema Matching and Mapping* (2011), 3–27. doi:10.1007/978-3-642-16518-4_1.
[104] L. Otero-Cerdeira, F.J. Rodríguez-Martínez and A. Gómez-Rodríguez, Ontology matching: A literature review, *Expert Systems with Applications* **42**(2) (2015). doi:10.1016/j.eswa.2014.08.032.
[105] H.-H. Do and E. Rahm, COMA—a system for flexible combination of schema matching approaches, in: *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, Elsevier, 2002, pp. 610–621.
[106] Y. Zhang, X. Wang, S. Lai, S. He, K. Liu, J. Zhao and X. Lv, Ontology Matching with Word Embeddings, in: *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data - 13th China National Conference, CCL 2014, and Second International Symposium, NLP-NABD 2014, Wuhan, China, October 18-19, 2014. Proceedings*, M. Sun, Y. Liu and J. Zhao, eds, Lecture Notes in Computer Science, Vol. 8801, Springer, 2014, pp. 34–45. doi:10.1007/978-3-319-12277-9_4.
[107] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 3111–3119. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.
[108] D. Ayala, I. Hernández, D. Ruiz and E. Rahm, LEAPME: Learning-based Property Matching with Embeddings, *Data & Knowledge Engineering* **137** (2022), 101943. doi:https://doi.org/10.1016/j.datak.2021.101943. https://www.sciencedirect.com/science/article/pii/S0169023X21000707.
[109] J. Portisch, G. Costa, K. Stefani, K. Kreplin, M. Hladik and H. Paulheim, Ontology Matching Through Absolute Orientation of Embedding Spaces, in: *The Semantic Web: ESWC 2022 Satellite Events - Hersonissos, Crete, Greece, May 29 - June 2, 2022, Proceedings*, P. Groth, A. Rula, J. Schneider, I. Tiddi, E. Simperl, P. Alexopoulos, R. Hoekstra, M. Alam, A. Dimou and M. Tamper, eds, Lecture Notes in Computer Science, Vol. 13384, Springer, 2022, pp. 153–157. doi:10.1007/978-3-031-11609-4_29.
[110] J. Portisch, M. Hladik and H. Paulheim, RDF2Vec Light - A Lightweight Approach for Knowledge Graph Embeddings, *CoRR* **abs/2009.07659** (2020). https://arxiv.org/abs/2009.07659.
[111] C.A. Knoblock, P.A. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan and P. Mallick, Semi-automatically Mapping Structured Sources into the Semantic Web, in: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, E. Simperl, P. Cimiano, A. Polleres, Ó. Corcho and V. Presutti, eds, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 375–390. doi:10.1007/978-3-642-30284-8_32.
[112] R.A. Pottinger and P.A. Bernstein, Merging models based on given correspondences, in: *Proceedings 2003 VLDB Conference*, Elsevier, 2003, pp. 862–873.
[113] S. Raunich and E. Rahm, Target-driven merging of taxonomies with ATOM, *Information Systems* **42** (2014), 1–14.
[114] I. Osman, S.B. Yahia and G. Diallo, Ontology integration: approaches and challenging issues, *Information Fusion* **71** (2021), 38–63.
[115] R. Usbeck, A.-C. Ngonga Ngomo, S. Auer, D. Gerber and A. Both, AGDISTIS - Graph-Based Disambiguation of Named Entities using Linked Data, in: *13th International Semantic Web Conference*, 2014. http://svn.aksw.org/papers/2014/ISWC_AGDISTIS/public.pdf.
[116] P. Ferragina and U. Scaiella, TAGME: on-the-fly annotation of short text fragments (by wikipedia entities), in: *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, J. Huang, N. Koudas, G.J.F. Jones, X. Wu, K. Collins-Thompson and A. An, eds, ACM, 2010, pp. 1625–1628. doi:10.1145/1871437.1871689.
[117] F. Piccinno and P. Ferragina, From TagME to WAT: A New Entity Annotator, in: *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 55–62–. ISBN 9781450330237. doi:10.1145/2633211.2634350.

[118] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard and D. McClosky, The Stanford CoreNLP Natural Language Processing Toolkit, in: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 55–60. doi:10.3115/v1/P14-5010. https://aclanthology.org/P14-5010.

[119] Y. Goldberg, *Neural Network Methods for Natural Language Processing*, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2017. doi:10.2200/S00762ED1V01Y201703HLT037.

[120] J. Li, A. Sun, J. Han and C. Li, A Survey on Deep Learning for Named Entity Recognition, *IEEE Trans. Knowl. Data Eng.* **34**(1) (2022), 50–70. doi:10.1109/TKDE.2020.2981314.

[121] S. Moon, L. Neves and V. Carvalho, Multimodal Named Entity Recognition for Short Social Media Posts, in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 852–860. doi:10.18653/v1/N18-1078. https://aclanthology.org/N18-1078.

[122] J. Yu, J. Jiang, L. Yang and R. Xia, Improving Multimodal Named Entity Recognition via Entity Span Detection with Unified Multimodal Transformer, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 3342–3352. doi:10.18653/v1/2020.acl-main.306. https://aclanthology.org/2020.acl-main.306.

[123] P. Pezeshkpour, L. Chen and S. Singh, Embedding Multimodal Relational Data for Knowledge Base Completion, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 3208–3218. doi:10.18653/v1/D18-1359. https://aclanthology.org/D18-1359.

[124] M. Li, A. Zareian, Y. Lin, X. Pan, S. Whitehead, B. Chen, B. Wu, H. Ji, S.-F. Chang, C. Voss, D. Napierski and M. Freedman, GAIA: A Fine-grained Multimedia Knowledge Extraction System, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, Online, 2020, pp. 77–86. doi:10.18653/v1/2020.acl-demos.11. https://aclanthology.org/2020.acl-demos.11.

[125] Y. Ding, J. Yu, B. Liu, Y. Hu, M. Cui and Q. Wu, MuKEA: Multimodal Knowledge Extraction and Accumulation for Knowledge-based Visual Question Answering, *CoRR* **abs/2203.09138** (2022).

[126] J.L. Martinez-Rodriguez, A. Hogan and I. Lopez-Arevalo, Information extraction meets the Semantic Web: A survey, *Semantic Web* **11**(2) (2020), 255–335. doi:10.3233/sw-180333. https://doi.org/10.3233%2Fsw-180333.

[127] S. Kulkarni, A. Singh, G. Ramakrishnan and S. Chakrabarti, Collective annotation of Wikipedia entities in web text, in: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, ACM Press, 2009. doi:10.1145/1557019.1557073. https://doi.org/10.1145%2F1557019.1557073.

[128] D. Milne and I.H. Witten, Learning to link with wikipedia, in: *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, ACM Press, 2008. doi:10.1145/1458082.1458150. https://doi.org/10.1145%2F1458082.1458150.

[129] X. Han, L. Sun and J. Zhao, Collective entity linking in web text: a graph-based method, in: *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, W. Ma, J. Nie, R. Baeza-Yates, T. Chua and W.B. Croft, eds, ACM, 2011, pp. 765–774. doi:10.1145/2009916.2010019.

[130] O. Medelyan, I.H. Witten and D. Milne, Topic Indexing with Wikipedia, in: *In Proc. of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08*, 2008.

[131] J. Hoffart, D. Milchevski, G. Weikum, A. Anand and J. Singh, The Knowledge Awakens: Keeping Knowledge Bases Fresh with Emerging Entities, in: *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks and B.Y. Zhao, eds, ACM, 2016, pp. 203–206. doi:10.1145/2872518.2890537.

[132] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute and V. Raghavendra, Deep Learning for Entity Matching: A Design Space Exploration, in: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, G. Das, C.M. Jermaine and P.A. Bernstein, eds, ACM, 2018, pp. 19–34. doi:10.1145/3183713.3196926.

[133] G. Zhou, M. Zhang, D.H. Ji and Q. Zhu, Tree Kernel-Based Relation Extraction with Context-Sensitive Structured Parse Tree Information, in: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 728–736. https://aclanthology.org/D07-1076.

[134] X. Han, T. Gao, Y. Lin, H. Peng, Y. Yang, C. Xiao, Z. Liu, P. Li, J. Zhou and M. Sun, More Data, More Relations, More Context and More Openness: A Review and Outlook for Relation Extraction, in: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, Association for Computational Linguistics, Suzhou, China, 2020, pp. 745–758. https://aclanthology.org/2020.aacl-main.75.

[135] S. Vashishth, P. Jain and P.P. Talukdar, CESI: Canonicalizing Open Knowledge Bases using Embeddings and Side Information, in: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F.L. Gandon, M. Lalmas and P.G. Ipeirotis, eds, ACM, 2018, pp. 1317–1327. doi:10.1145/3178876.3186030.

[136] J. Daiber, M. Jakob, C. Hokamp and P.N. Mendes, Improving Efficiency and Accuracy in Multilingual Entity Extraction, in: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[137] R. Clancy, I.F. Ilyas and J. Lin, Scalable Knowledge Graph Construction from Text Collections, in: *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 39–46. doi:10.18653/v1/D19-6607. https://aclanthology.org/D19-6607.

[138] X. Han, T. Gao, Y. Yao, D. Ye, Z. Liu and M. Sun, OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 169–174. doi:10.18653/v1/D19-3029. https://aclanthology.org/D19-3029.

[139] D. Elliott and F. Keller, Image Description using Visual Dependency Representations, in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1292–1302. https://aclanthology.org/D13-1128.

[140] C. Zheng, J. Feng, Z. Fu, Y. Cai, Q. Li and T. Wang, Multimodal Relation Extraction with Efficient Graph Alignment, in: *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 5298–5306–. ISBN 9781450386517. doi:10.1145/3474085.3476968.

[141] H. Köpcke and E. Rahm, Frameworks for entity matching: A comparison, *Data & Knowledge Engineering* **69**(2) (2010), 197–210.

[142] P. Christen, The data matching process, in: *Data matching*, Springer, 2012, pp. 23–35.

[143] N. M., H. M., N.N. A. and R. E., A survey of current link discovery frameworks, *Semantic Web* **8**(3) (2017).

[144] N. Barlaug and J.A. Gulla, Neural networks for entity matching: A survey, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **15**(3) (2021), 1–37.

[145] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis and K. Stefanidis, An Overview of End-to-End Entity Resolution for Big Data, *ACM Computing Surveys* (2020).

[146] G. Papadakis, D. Skoutas, E. Thanos and T. Palpanas, Blocking and filtering techniques for entity resolution: A survey, *ACM Computing Surveys (CSUR)* **53**(2) (2020), 1–42.

[147] A. Saeedi, E. Peukert and E. Rahm, Using link features for entity clustering in knowledge graphs, in: *European Semantic Web Conference*, Springer, 2018, pp. 576–592.

[148] G. Papadakis, L. Tsekouras, E. Thanos, N. Pittaras, G. Simonini, D. Skoutas, P. Isaris, G. Giannakopoulos, T. Palpanas and M. Koubarakis, JedAI3: beyond batch, blocking-based Entity Resolution., in: *EDBT*, 2020, pp. 603–606.

[149] M. Ebraheem, S. Thirumuruganathan, S.R. Joty, M. Ouzzani and N. Tang, DeepER - Deep Entity Resolution, *ArXiv preprint* **abs/1710.00597** (2017). https://arxiv.org/abs/1710.00597.

[150] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami and C. Li, A Benchmarking Study of Embedding-based Entity Alignment for Knowledge Graphs, *Proc. VLDB Endow.* **13**(11) (2020), 2326–2340. http://www.vldb.org/pvldb/vol13/p2326-sun.pdf.

[151] D. Obraczka, J. Schuchart and E. Rahm, Embedding-Assisted Entity Resolution for Knowledge Graphs, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*, D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna and J.F. Sequeda, eds, CEUR Workshop Proceedings, Vol. 2873, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2873/paper8.pdf.

[152] M. Leone, S. Huber, A. Arora, A. García-Durán and R. West, A Critical Re-evaluation of Neural Methods for Entity Alignment, *Proc. VLDB Endow.* **15**(8) (2022), 1712–1725. https://www.vldb.org/pvldb/vol15/p1712-arora.pdf.

[153] G. Papadakis, E. Ioannou, E. Thanos and T. Palpanas, *The Four Generations of Entity Resolution*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2021. doi:10.2200/S01067ED1V01Y202012DTM064.

[154] A. Gruenheid, X.L. Dong and D. Srivastava, Incremental record linkage, *Proceedings of the VLDB Endowment* **7**(9) (2014), 697–708.

[155] L. Gazzarri and M. Herschel, End-to-end Task Based Parallelization for Entity Resolution on Dynamic Data, in: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, IEEE, 2021, pp. 1248–1259.

[156] A. Saeedi, M. Nentwig, E. Peukert and E. Rahm, Scalable matching and clustering of entities with FAMER, *Complex Systems Informatics and Modeling Quarterly* **16** (2018), 61–83.

[157] B. Ramadan and P. Christen, Forest-Based Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution, in: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 1787–1790–. ISBN 9781450325981. doi:10.1145/2661829.2661869.

[158] B. Ramadan, P. Christen, H. Liang and R.W. Gayler, Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution, *J. Data and Information Quality* **6**(4) (2015). doi:10.1145/2816821.

[159] D. Karapiperis, A. Gkoulalas-Divanis and V.S. Verykios, Summarization Algorithms for Record Linkage., in: *EDBT*, 2018, pp. 73–84.

[160] T. Brasileiro Araújo, K. Stefanidis, C.E. Santos Pires, J. Nummenmaa and T. Pereira da Nóbrega, Incremental blocking for entity resolution over web streaming data, in: *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019, pp. 332–336.

[161] T.B. Araújo, K. Stefanidis, C.E. Santos Pires, J. Nummenmaa and T.P. Da Nóbrega, Schema-agnostic blocking for streaming data, in: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 412–419.

[162] O. Hassanzadeh, F. Chiang, H.C. Lee and R.J. Miller, Framework for evaluating clustering algorithms in duplicate detection, *Proc. of the VLDB Endowment* **2**(1) (2009), 1282–1293.

[163] A. Saeedi, E. Peukert and E. Rahm, Comparative evaluation of distributed clustering schemes for multi-source entity resolution, in: *European Conference on Advances in Databases and Information Systems*, Springer, 2017, pp. 278–293.

[164] W. M., S. A. and D. C., Fast and accurate incremental entity resolution relative to an entity knowledge base, in: *CIKM*, 2012.

[165] J. Bleiholder and F. Naumann, Data fusion, *ACM computing surveys (CSUR)* **41**(1) (2009), 1–41.

[166] C. Bizer, C. Becker, P.N. Mendes, R. Isele, A. Matteini and A. Schultz, Ldif-a framework for large-scale linked data integration (2012).

[167] P.N. Mendes, H. Mühleisen and C. Bizer, Sieve: linked data quality assessment and fusion, in: *Proceedings of the 2012 joint EDBT/ICDT workshops*, 2012, pp. 116–123.

[168] X. Dong, L. Berti-Équille and D. Srivastava, Data Fusion: Resolving Conflicts from Multiple Sources, in: *Interational Conference on Web-Age Information Management*, 2013.

[169] R.Y. Wang and D.M. Strong, Beyond Accuracy: What Data Quality Means to Data Consumers, *J. Manage. Inf. Syst.* **12**(4) (1996), 5–33–. doi:10.1080/07421222.1996.11518099.

[170] C. Bizer and R. Cyganiak, Quality-driven information filtering using the WIQA policy framework, *Journal of Web Semantics* **7**(1) (2009), 1–10.

[171] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer and J. Lehmann, Crowdsourcing linked data quality assessment, in: *International semantic web conference*, Springer, 2013, pp. 260–276.

[172] F. Li, X.L. Dong, A. Langen and Y. Li, Knowledge verification for long-tail verticals, *Proceedings of the VLDB Endowment* **10**(11) (2017), 1370–1381.

[173] J. Lehmann, D. Gerber, M. Morsey and A.-C. Ngonga Ngomo, Defacto-deep fact validation, in: *International semantic web conference*, Springer, 2012, pp. 312–327.

[174] H. Paulheim and C. Bizer, Type Inference on Noisy RDF Data, in: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J.X. Parreira, L. Aroyo, N.F. Noy, C. Welty and K. Janowicz, eds, Lecture Notes in Computer Science, Vol. 8218, Springer, 2013, pp. 510–525. doi:10.1007/978-3-642-41335-3_32.

[175] H. Paulheim and C. Bizer, Improving the Quality of Linked Data Using Statistical Distributions, *Int. J. Semantic Web Inf. Syst.* **10**(2) (2014), 63–86. doi:10.4018/ijswis.2014040104.

[176] A. Lutov, S. Roshankish, M. Khayati and P. Cudré-Mauroux, StaTIX - Statistical Type Inference on Linked Data, in: *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, N. Abe, H. Liu, C. Pu, X. Hu, N.K. Ahmed, M. Qiao, Y. Song, D. Kossmann, B. Liu, K. Lee, J. Tang, J. He and J.S. Saltz, eds, IEEE, 2018, pp. 2253–2262. doi:10.1109/BigData.2018.8622285.

[177] Y. Zhao, A. Zhang, R. Xie, K. Liu and X. Wang, Connecting Embeddings for Knowledge Graph Entity Typing, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 6419–6428. doi:10.18653/v1/2020.acl-main.572. https://aclanthology.org/2020.acl-main.572.

[178] A.P. Aprosio, C. Giuliano and A. Lavelli, Extending the Coverage of DBpedia Properties using Distant Supervision over Wikipedia., in: *NLP-DBPEDIA@ ISWC*, Citeseer, 2013.

[179] D. Gerber, S. Hellmann, L. Bühmann, T. Soru, R. Usbeck and A.-C. Ngonga Ngomo, Real-time RDF extraction from unstructured data streams, in: *International semantic web conference*, Springer, 2013, pp. 135–150.

[180] D. Gerber and A.-C.N. Ngomo, Bootstrapping the linked data web, in: *1st Workshop on Web Scale Knowledge Extraction@ ISWC*, Vol. 2011, 2011, p. 61.

[181] M. Mintz, S. Bills, R. Snow and D. Jurafsky, Distant supervision for relation extraction without labeled data, in: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Association for Computational Linguistics, Suntec, Singapore, 2009, pp. 1003–1011. https://aclanthology.org/P09-1113.

[182] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta and D. Lin, Knowledge base completion via search-based question answering, in: *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, C. Chung, A.Z. Broder, K. Shim and T. Suel, eds, ACM, 2014, pp. 515–526. doi:10.1145/2566486.2568032.

[183] D. Lange, C. Böhm and F. Naumann, Extracting structured information from Wikipedia articles to populate infoboxes, in: *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, J. Huang, N. Koudas, G.J.F. Jones, X. Wu, K. Collins-Thompson and A. An, eds, ACM, 2010, pp. 1661–1664. doi:10.1145/1871437.1871698.

[184] C.R. Fields, Probabilistic models for segmenting and labeling sequence data, in: *ICML 2001*, 2001.

[185] T. Blevins and L. Zettlemoyer, Moving Down the Long Tail of Word Sense Disambiguation with Gloss Informed Bi-encoders, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 1006–1017. doi:10.18653/v1/2020.acl-main.95. https://aclanthology.org/2020.acl-main.95.

[186] E. Munoz, A. Hogan and A. Mileo, Triplifying wikipedia's tables, *LD4IE@ ISWC* (2013).

[187] D. Ritze, O. Lehmberg and C. Bizer, Matching html tables to dbpedia, in: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, 2015, pp. 1–6.

[188] H. Paulheim and S.P. Ponzetto, Extending DBpedia with Wikipedia List Pages., *NLP-DBPEDIA@ ISWC* **13** (2013).

[189] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[190] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge Graph Embedding by Translating on Hyperplanes, in: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, C.E. Brodley and P. Stone, eds, AAAI Press, 2014, pp. 1112–1119. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[191] P. Kolyvakis, A. Kalousis and D. Kiritsis, HyperKG: Hyperbolic Knowledge Graph Embeddings for Knowledge Base Completion, *ArXiv preprint* **abs/1908.04895** (2019). https://arxiv.org/abs/1908.04895.

[192] M. Ali, M. Berrendorf, C.T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp and J. Lehmann, Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1. doi:10.1109/TPAMI.2021.3124805.

[193] K.K. Teru, E.G. Denis and W.L. Hamilton, Inductive Relation Prediction by Subgraph Reasoning, in: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, Vol. 119, PMLR, 2020, pp. 9448–9457. http://proceedings.mlr.press/v119/teru20a.html.

[194] M. Galkin, E. Denis, J. Wu and W.L. Hamilton, NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs, in: *International Conference on Learning Representations*, 2022. https://openreview.net/forum?id=xMJWUKJnFSw.

[195] 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.

[196] X. Dong, X. He, A. Kan, X. Li, Y. Liang, J. Ma, Y. Xu, C. Zhang, T. Zhao, G.B. Saldana, S. Deshpande, A.M. Manduca, J. Ren, S.P. Singh, F. Xiao, H.-S. Chang, G. Karamanolakis, Y. Mao, Y. Wang, C. Faloutsos, A. McCallum and J. Han, AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types, *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).

[197] K. Bollacker, R. Cook and P. Tufts, Freebase: A shared database of structured general human knowledge, in: *AAAI*, Vol. 7, 2007, pp. 1962–1963.

[198] M. Hofer, S. Hellmann, M. Dojchinovski and J. Frey, The new dbpedia release cycle: Increasing agility and efficiency in knowledge extraction workflows, in: *International Conference on Semantic Systems*, Springer, Cham, 2020, pp. 1–18.

[199] F.M. Suchanek, G. Kasneci and G. Weikum, Yago: a core of semantic knowledge, in: *The Web Conference*, 2007.

[200] T. Pellissier Tanon, G. Weikum and F. Suchanek, Yago 4: A reason-able knowledge base, in: *European Semantic Web Conference*, Springer, 2020, pp. 583–596.

[201] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R.H. Jr. and T.M. Mitchell, Toward an Architecture for Never-Ending Language Learning, in: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, M. Fox and D. Poole, eds, AAAI Press, 2010. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1879.

[202] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85.

[203] M. Morsey, J. Lehmann, S. Auer, C. Stadler and S. Hellmann, Dbpedia and the live extraction of structured data from wikipedia, *Program* (2012).

[204] G. Gawriljuk, A. Harth, C.A. Knoblock and P. Szekely, A scalable approach to incrementally building knowledge graphs, in: *International Conference on Theory and Practice of Digital Libraries*, Springer, 2016, pp. 188–199.

[205] S. Auer, A. Oelen, M. Haris, M. Stocker, J. D'Souza, K.E. Farfar, L. Vogt, M. Prinz, V. Wiens and M.Y. Jaradeh, Improving access to scientific literature with knowledge graphs, *Bibliothek Forschung und Praxis* **44**(3) (2020), 516–529.

[206] D. Dessì, F. Osborne, D. Reforgiato Recupero, D. Buscaldi, E. Motta and H. Sack, Ai-kg: an automatically generated knowledge graph of artificial intelligence, in: *International Semantic Web Conference*, Springer, 2020, pp. 127–143.

[207] M. Preusse, A. Jarasch, T. Bleimehl, S. Muller, J. Munro, L. Gutebier, R. Henkel and D. Waltemath, COVIDGraph: Connecting Biomedical COVID-19 Resources and Computational Biology Models, in: *Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021) co-located with 47th International Conference on Very Large Data Bases (VLDB 2021), Copenhagen, Denmark, August 20, 2021*, D. Mottin, M. Lissandrini, S.B. Roy and Y. Velegrakis, eds, CEUR Workshop Proceedings, Vol. 2929, CEUR-WS.org, 2021, pp. 34–37. http://ceur-ws.org/Vol-2929/paper6.pdf.

[208] V.N. Ioannidis, X. Song, S. Manchanda, M. Li, X. Pan, D. Zheng, X. Ning, X. Zeng and G. Karypis, DRKG - Drug Repurposing Knowledge Graph for Covid-19, 2020.

[209] H. Alberts, T. Huang, Y. Deshpande, Y. Liu, K. Cho, C. Vania and I. Calixto, VisualSem: a high-quality knowledge graph for vision and language, *ArXiv* **abs/2008.09150** (2020).

[210] A. Dsouza, N. Tempelmeier, R. Yu, S. Gottschalk and E. Demidova, WorldKG: A World-Scale Geographic Knowledge Graph, *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).

[211] Y. Zhang, M. Sheng, R. Zhou, Y. Wang, G. Han, H. Zhang, C. Xing and J. Dong, HKGB: An Inclusive, Extensible, Intelligent, Semi-auto-constructed Knowledge Graph Framework for Healthcare with Clinicians' Expertise Incorporated, *Inf. Process. Manag.* **57** (2020), 102324.

[212] A. Ekelhart, F.J. Ekaputra and E. Kiesling, The SLOGERT Framework for Automated Log Knowledge Graph Construction, in: *ESWC*, 2021.

[213] M. Galkin, S. Auer, M.-E. Vidal and S. Scerri, Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems, in: *Proceedings of the 19th International Conference on Enterprise Information Systems - Volume 2: ICEIS,*, SciTePress, 2017, pp. 88–98, INSTICC. ISBN 978-989-758-248-6. doi:10.5220/0006325200880098.

[214] M. Färber, The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data, in: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I.F. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Lecture Notes in Computer Science, Vol. 11779, Springer, 2019, pp. 113–129. doi:10.1007/978-3-030-30796-7_8.

[215] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner and L. Pintscher, From freebase to wikidata: The great migration, in: *Proceedings of the 25th international conference on world wide web*, 2016, pp. 1419–1428.

[216] A. Piscopo, L.-A. Kaffee, C. Phethean and E. Simperl, Provenance information in a collaborative knowledge graph: an evaluation of Wikidata external references, in: *International semantic web conference*, Springer, 2017, pp. 542–558.

[217] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z.G. Ives, DBpedia: A Nucleus for a Web of Open Data, in: *ISWC/ASWC*, 2007.

[218] A. Hofmann, S. Perchani, J. Portisch, S. Hertling and H. Paulheim, DBkWik: Towards Knowledge Graph Creation from Thousands of Wikis., in: *ISWC (Posters, Demos & Industry Tracks)*, 2017.

[219] S. Hellmann, C. Stadler, J. Lehmann and S. Auer, DBpedia Live Extraction, in: *OTM Conferences*, 2009.

[220] C. Fellbaum (ed.), *WordNet An Electronic Lexical Database*, The MIT Press, Cambridge, MA ; London, 1998. ISBN 978-0-262-06197-1. http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8106.

[221] A.J.G. Gray, C.A. Goble and R. Jimenez, Bioschemas: From Potato Salad to Protein Annotation, in: *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, N. Nikitina, D. Song, A. Fokoue and P. Haase, eds, CEUR Workshop Proceedings, Vol. 1963, CEUR-WS.org, 2017. http://ceur-ws.org/Vol-1963/paper579.pdf.

[222] J.M. Giménez-García, M.C. Duarte, A. Zimmermann, C. Gravier, E.R. Hruschka and P. Maret, NELL2RDF: Reading the Web, Tracking the Provenance, and Publishing it as Linked Data, in: *CKGSemStats@ISWC*, 2018.

[223] A.A. Salatino, T. Thanapalasingam, A. Mannocci, F. Osborne and E. Motta, The Computer Science Ontology: A Large-Scale Taxonomy of Research Areas, in: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*, D. Vrandecic, K. Bontcheva, M.C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee and E. Simperl, eds, Lecture Notes in Computer Science, Vol. 11137, Springer, 2018, pp. 187–205. doi:10.1007/978-3-030-00668-6_12.

[224] D. Wadden, U. Wennberg, Y. Luan and H. Hajishirzi, Entity, Relation, and Event Extraction with Contextualized Span Representations, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng and X. Wan, eds, Association for Computational Linguistics, 2019, pp. 5783–5788. doi:10.18653/v1/D19-1585.

[225] A.A. Salatino, F. Osborne, T. Thanapalasingam and E. Motta, The CSO Classifier: Ontology-Driven Detection of Research Topics in Scholarly Articles, in: *Digital Libraries for Open Knowledge - 23rd International Conference on Theory and Practice of Digital Libraries, TPDL 2019, Oslo, Norway, September 9-12, 2019, Proceedings*, A. Doucet, A. Isaac, K. Golub, T. Aalberg and A. Jatowt, eds, Lecture Notes in Computer Science, Vol. 11799, Springer, 2019, pp. 296–311. doi:10.1007/978-3-030-30760-8_26.

[226] D. Dessì, F. Osborne, D.R. Recupero, D. Buscaldi and E. Motta, CS-KG: A Large-Scale Knowledge Graph of Research Entities and Claims in Computer Science, in: *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings*, U. Sattler, A. Hogan, C.M. Keet, V. Presutti, J.P.A. Almeida, H. Takeda, P. Monnin, G. Pirrò and C. d'Amato, eds, Lecture Notes in Computer Science, Vol. 13489, Springer, 2022, pp. 678–696. doi:10.1007/978-3-031-19433-7_39.

[227] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C.H. So and J. Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics* **36** (2019), 1234–1240.

[228] K.D. Pruitt, T.A. Tatusova and D.R. Maglott, NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins, *Nucleic Acids Res.* **35**(Database–Issue) (2007), 61–65. doi:10.1093/nar/gkl842.

[229] B. Steenwinckel, G. Vandewiele, I. Rausch, P. Heyvaert, P. Colpaert, P. Simoens, A. Dimou, F.D. Turkc and F. Ongenae, Facilitating COVID-19 Meta-analysis Through a Literature Knowledge Graph, in: *Accepted in Proc. of 19th International Semantic Web Conference (ISWC)*, 2020.

[230] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg and L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)* **115**(3) (2015), 211–252. doi:10.1007/s11263-015-0816-y.

[231] H. Alberts and I. Calixto, ImagiFilter: A resource to enable the semi-automatic mining of images at scale, *ArXiv* **abs/2008.09152** (2020).

[232] A. Dsouza, N. Tempelmeier and E. Demidova, Towards Neural Schema Alignment for OpenStreetMap and Knowledge Graphs, in: *SEMWEB*, 2021.

[233] G. Demartini, D.E. Difallah and P. Cudré-Mauroux, ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking, in: *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, A. Mille, F. Gandon, J. Misselis, M. Rabinovich and S. Staab, eds, ACM, 2012, pp. 469–478. doi:10.1145/2187836.2187900.

[234] A. Tonon, M. Catasta, R. Prokofyev, G. Demartini, K. Aberer and P. Cudré-Mauroux, Contextualized ranking of entity types based on knowledge graphs, *Journal of Web Semantics* **37** (2016), 170–183.

[235] K. Aberer, A. Boyarsky, P. Cudré-Mauroux, G. Demartini and O. Ruchayskiy, Sciencewise: A web-based interactive semantic platform for scientific collaboration, in: *10th International Semantic Web Conference (ISWC 2011-Demo), Bonn, Germany*, 2011.

[236] A. Tonon, P. Cudré-Mauroux, A. Blarer, V. Lenders and B. Motik, ArmaTweet: detecting events by semantic tweet analysis, in: *European Semantic Web Conference*, Springer, 2017, pp. 138–153.

[237] R. Mavlyutov, C. Curino, B. Asipov and P. Cudré-Mauroux, Dependency-Driven Analytics: A Compass for Uncharted Data Oceans, in: *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*, www.cidrdb.org, 2017. http://cidrdb.org/cidr2017/papers/p59-mavlyutov-cidr17.pdf.

[238] G. Zheng, S. Mukherjee, X. Dong and F. Li, OpenTag: Open Attribute Value Extraction from Product Profiles, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018).

[239] K. Kurniawan, A. Ekelhart, E. Kiesling, A. Froschl and F.J. Ekaputra, Semantic Integration and Monitoring of File System Activity, in: *Proceedings of the Posters and Demo Track of the 15th International Conference on Semantic Systems co-located with 15th International Conference on Semantic Systems (SEMANTiCS 2019), Karlsruhe, Germany, September 9th - to - 12th, 2019*, M. Alam, R. Usbeck, T. Pellegrini, H. Sack and Y. Sure-Vetter, eds, CEUR Workshop Proceedings, Vol. 2451, CEUR-WS.org, 2019. http://ceur-ws.org/Vol-2451/paper-17.pdf.

[240] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov, Silk - A Link Discovery Framework for the Web of Data, in: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*, C. Bizer, T. Heath, T. Berners-Lee and K. Idehen, eds, CEUR Workshop Proceedings, Vol. 538, CEUR-WS.org, 2009. http://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.

[241] X. Pan, D.S. Papailiopoulos, S. Oymak, B. Recht, K. Ramchandran and M.I. Jordan, Parallel Correlation Clustering on Big Graphs, in: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama and R. Garnett, eds, 2015, pp. 82–90. https://proceedings.neurips.cc/paper/2015/hash/b53b3a3d6ab90ce0268229151c9bde11-Abstract.html.

[242] P. Bhattarai, M. Ghassemi and T. Alhanai, Open-Source Code Repository Attributes Predict Impact of Computer Science Research, in: *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, JCDL '22, Association for Computing Machinery, New York, NY, USA, 2022. ISBN 9781450393454. doi:10.1145/3529372.3530927.

[243] M. Mahdavi, F. Neutatz, L. Visengeriyeva and Z. Abedjan, Towards automated data cleaning workflows, *Machine Learning* **15** (2019), 16.

[244] X. Zhao, Y. Jia, A. Li, R. Jiang and Y. Song, Multi-source knowledge fusion: a survey, *World Wide Web* **23**(4) (2020), 2567–2592. doi:10.1007/s11280-020-00811-0.

[245] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe and P.A. Szekely, A study of the quality of Wikidata, *J. Web Semant.* **72** (2022), 100679. doi:10.1016/j.websem.2021.100679.

[246] A.G. Nuzzolese, A.L. Gentile, V. Presutti, A. Gangemi, D. Garigliotti and R. Navigli, Open Knowledge Extraction Challenge, in: *SemWebEval@ESWC*, 2015.

[247] J.M. Rodríguez, H.D. Merlino, P. Pesado and R. García-Martínez, Performance Evaluation of Knowledge Extraction Methods, in: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2016.

[248] Y. Zhang, V. Zhong, D. Chen, G. Angeli and C.D. Manning, Position-aware Attention and Supervised Data Improve Slot Filling, in: *Conference on Empirical Methods in Natural Language Processing*, 2017.

[249] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko and C.T. dos Santos, Ontology Alignment Evaluation Initiative: Six Years of Experience, *J. Data Semant.* **15** (2011), 158–192.

[250] H. Köpcke, A. Thor and E. Rahm, Evaluation of entity resolution approaches on real-world match problems, *Proc. VLDB Endow.* **3**(1) (2010), 484–493. doi:10.14778/1920841.1920904. http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/E04.pdf.

[251] S. Hertling and H. Paulheim, Gollum: A Gold Standard for Large Scale Multi Source Knowledge Graph Matching, *arXiv preprint arXiv:2209.07479* (2022).

[252] M. Galkin, M. Berrendorf and C.T. Hoyt, An Open Challenge for Inductive Link Prediction on Knowledge Graphs, *CoRR* **abs/2203.01520** (2022). doi:10.48550/arXiv.2203.01520.

[253] T. Safavi and D. Koutra, CoDEx: A Comprehensive Knowledge Graph Completion Benchmark, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, 2020, pp. 8328–8350. doi:10.18653/v1/2020.emnlp-main.669.

[254] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong and J. Leskovec, OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs, in: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, eds, 2021. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/db8e1af0cb3aca1ae2d0018624204529-Abstract-round2.html.

[255] J. Portisch, M. Hladik and H. Paulheim, Background knowledge in ontology matching: A survey, *Semantic Web* (2022).

[256] M. Röder, D. Kuchelev and A.N. Ngomo, HOBBIT: A platform for benchmarking Big Linked Data, *Data Sci.* **3**(1) (2020), 15–35. doi:10.3233/ds-190021.

[257] I.L. Oliveira, R. Fileto, R. Speck, L.P.F. Garcia, D. Moussallem and J. Lehmann, Towards holistic Entity Linking: Survey and directions, *Information Systems* **95** (2021), 101624. doi:https://doi.org/10.1016/j.is.2020.101624. https://www.sciencedirect.com/science/article/pii/S0306437920300958.