## Exploiting Linked Data and Knowledge Graphs in Large Organizations

**Part I Background**

**Part II Techniques**

**Part III Users Cases**

**1. Enterprise Knowledge Graph: An Introduction**

* 1. **A Brief History of Knowledge Graph**
  2. **Knowledge Graph Technologies in a Nutshell**
  3. **Applications of Knowledge Graphs for Enterprise**
  4. **How to Read This Book**

**Part I Knowledge Graph Foundations & Architecture**

1. **Knowledge Graph Foundations** 
   1. **Knowledge Representation and Query Languages (RDF, OWL, SPARQL)**
   2. **Ontologies and Vocabularies ...**
   3. **Data Lifting Standards (RDB2RDF)**
   4. **KG & Linked data**
   5. **For Web search and for Enterprise**
2. **Knowledge Architecture for Organizations**
   1. **Overview**
   2. **Acquisition and Integration (Ontology, NER, Ontology learning)**
   3. **Storing and Accessing Layer (RDF stores, Property graph-based)**
   4. **Consumption layer (Semantic search, Query Generation and Answering)**

**Part II Constructing, Understanding and Consuming Knowledge Graphs**

1. **Construction of Enterprise KG (I)**
   1. **Construction and maintenance lifecycle**
   2. **Ontology Authoring (Question-Driven approach)**
   3. **Semi-automated Linking of enterprise data for virtual KG (Semantic tagging and data linking)**

**5. Construction of Enterprise Knowledge Graphs (II)\***

**5.1 Named Entity, Thematic scope**

**5.2 Schema learning for KG**

**6. Understanding Knowledge Graphs**

**6.1 The Things in KG**

**6.2 Entity description**

**6.3 Profiling KG: Summarization**

**6.4 Query Generation**

**7. QA in KG**

**7.1 Over Text Doc.**

**7.2 QA over KG**

**7.3 Waston DeepQA**

**Part III Industrial Applications and Successful Stories**

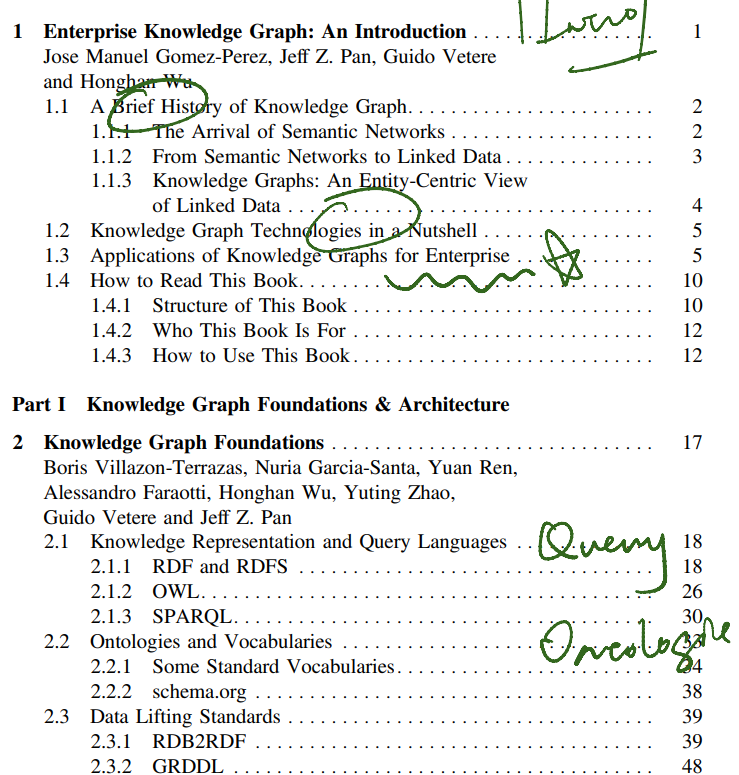
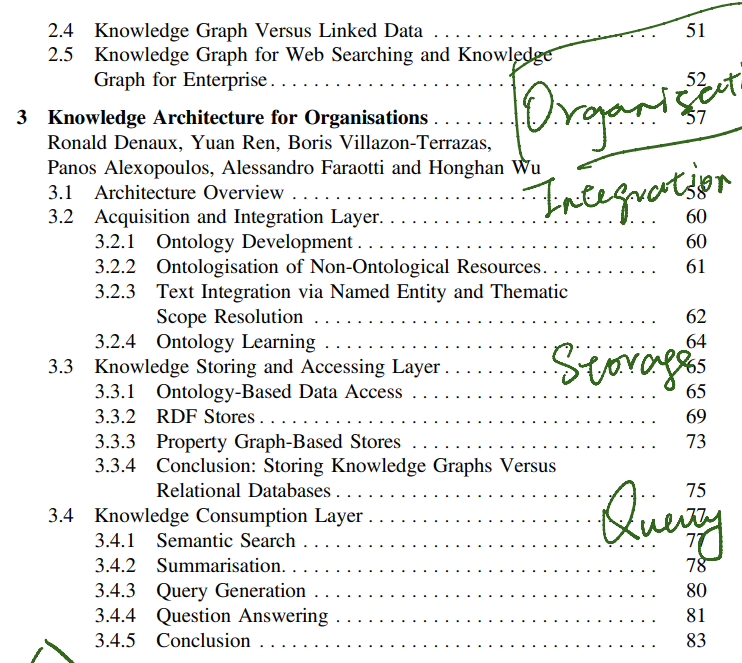
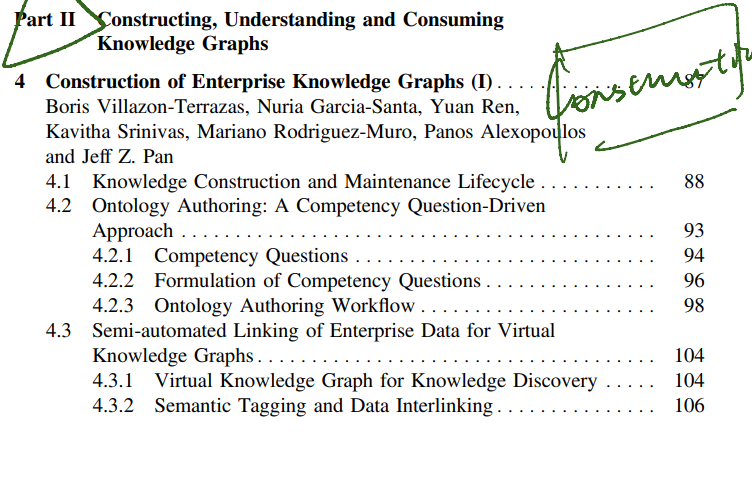
**8 Success Stories**

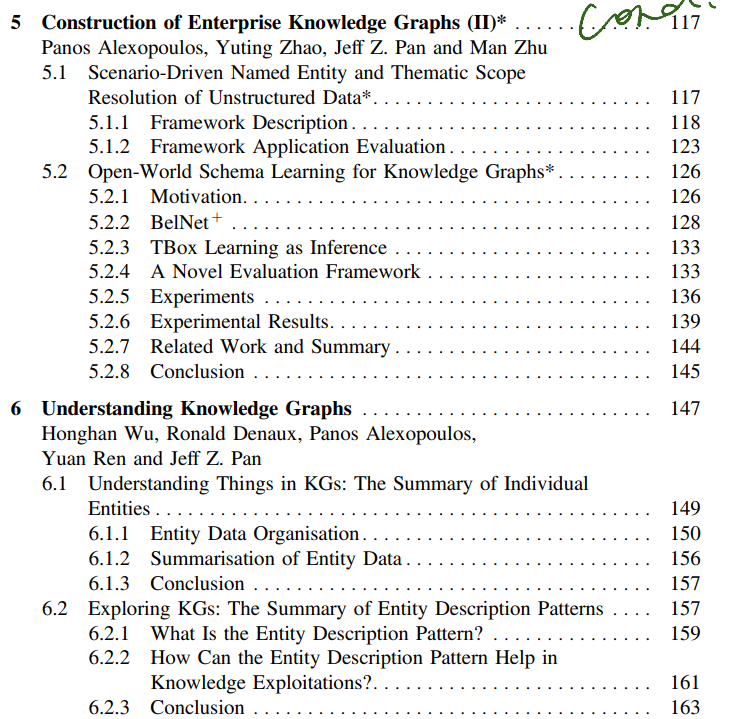
**8.1 IN the Media Industry**

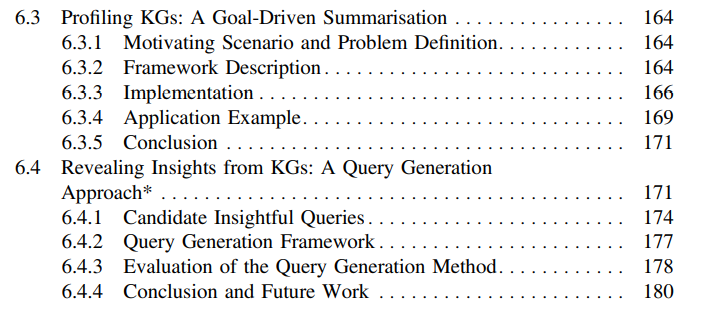
**8.2 In Cultural Heritage**

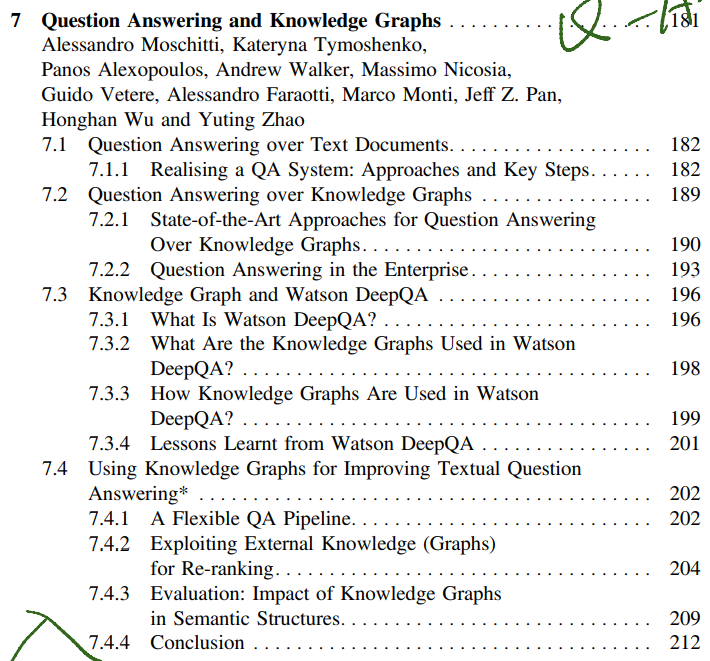
**8.3 In healthcare**

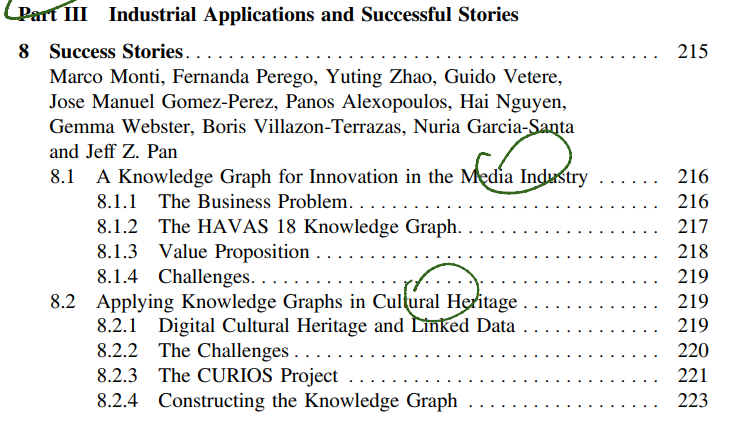
**9 The future**

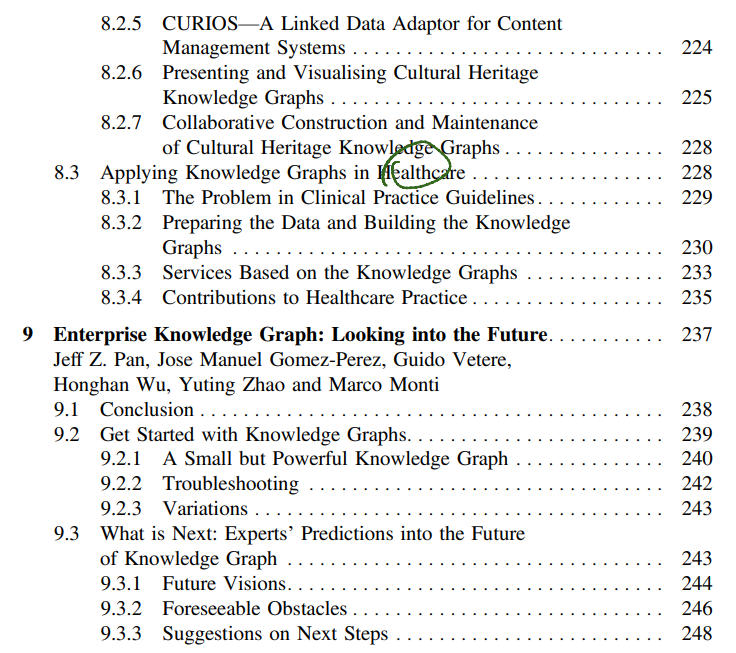
  












**1. Enterprise Knowledge Graph: An Introduction**

* 1. **A Brief History of Knowledge Graph**

The basic unit of a knowledge graph is (the representation of) a singular entity, such as a football match you are watching, a city you will visit soon or anything you would like to describe. Each entity might have various attributes. For example, the attributes of a person include name, birthdate, nationality, etc. Furthermore, entities are connected to each other by relations; e.g. you follow one of your colleagues in Twitter. Relations can be used to bridge two separate knowledge graphs.

Types of entities and relations are defined in some machine-understandable dictionaries called ontologies. The standard ontology language is called OWL (Web Ontology Language).

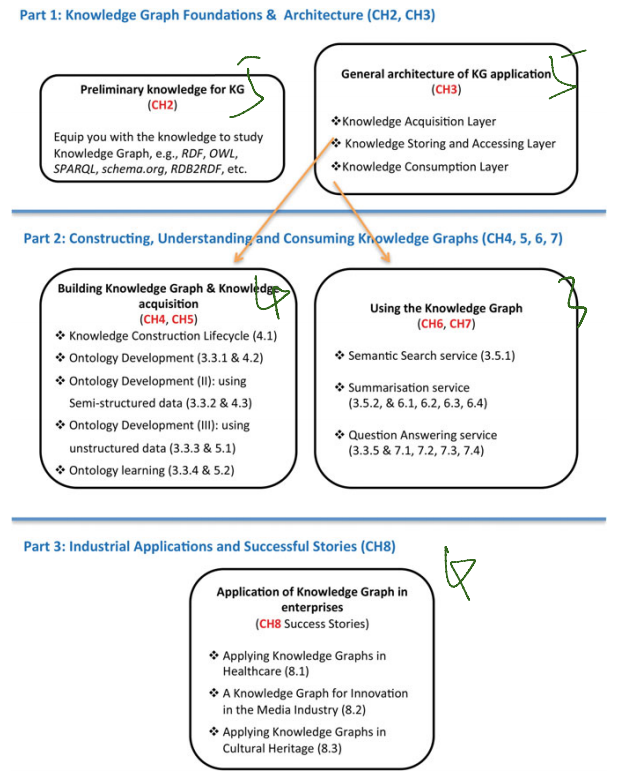
One of the first computer-based knowledge representation approaches are Semantic Networks, which represent knowledge in the form of interconnected nodes and arcs, where nodes represent objects, concepts or situations, and edges represent the relations between them, including is-a (e.g. “a chair is a type of furniture”) and part-of (e.g. “a seat is part of a chair”). Furthermore, Semantic Networks do not allow users to define the meaning of labels on nodes and arcs.

Based on RDF and OWL, Linked Data is a common framework to publish and share data across different applications and domains, where RDF provides a graph-based data model to describe objects. OWL offers a standard way of defining vocabularies for data annotations. In the Linked Data paradigm.

* 1. **Knowledge Graph Technologies in a Nutshell**

In many other cases, a knowledge graph can enhance the effectiveness of a traditional information processing/access task (e.g. information extraction, search, recommendation, question answering, etc.) by providing a valuable background domain knowledge.

* 1. **Applications of Knowledge Graphs for Enterprise**
  2. **How to Read This Book**



Paragraph description in P29

**Part I Knowledge Graph Foundations & Architecture**

1. **Knowledge Graph Foundations** 
   1. **Knowledge Representation and Query Languages (RDF, OWL, SPARQL)**



The blank-node

RDF provides the standard data model for a knowledge graph. There are several serialisation syntaxes for storing and exchanging RDF, such as Turtle [190], RDF/XML [3], RDFa [4], N-Triples [2], NQUADS [1] and JSON-LD [220].

**RDFS** provides a simple schema language for RDF, and allows one to declare classes/properties, using the predefined language level class rdfs:Class/property rdfs:Property; see the following example of the declaring org:Organization as a class and org:hasHomePage as a property: (See limitation, not powerful)

While RDF is the modern standard for Knowledge Graph/Semantic Network, the standard ontology language OWL is based on a family of formal knowledge representations called Description Logics (DLs) [16]. Syntactically **OWL** can be regarded as an extension of RDFS with additional vocabulary predefined by the OWL schema.3 This schema vocabulary provides extensively high expressive power for people to construct ontologies and/or to annotate their data, such as qualified cardinality restriction, property chain, self-restriction, symmetric and/or reflexive property.

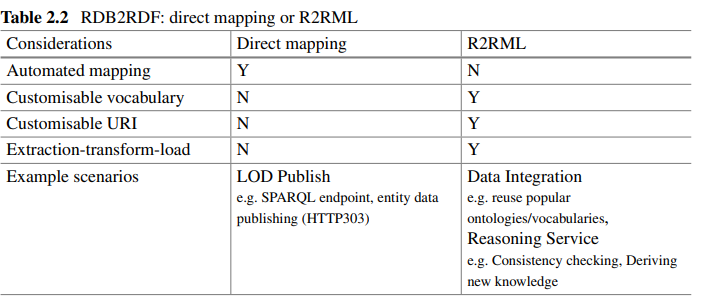
Now that we have RDF and OWL for constructing knowledge graphs and their schemas, we will introduce the standard query language for RDF and OWL. **SPARQL** statements are expressed according to the Turtle syntax,10 smoother than XML, and are based on the pattern-matching mechanism. The basic fragment of an SPARQL query resembles an RDF triple

* 1. **Ontologies and Vocabularies**
  2. **Data Lifting Standards (RDB2RDF)**

In many large organizations, the data or knowledge might take various formats, such as relational databases, Web pages, documentations, transaction logs, etc. To make this information accessible in the organization’s knowledge graph, it requires to convert them from their current representation into the format of knowledge representation. In our scenario, it is the RDF data model. The conversion process is called data lifting, which means the conversion is not only a transform from one format to another, but also a “lift” of the information from the data level into the machine-readable “knowledge” level.

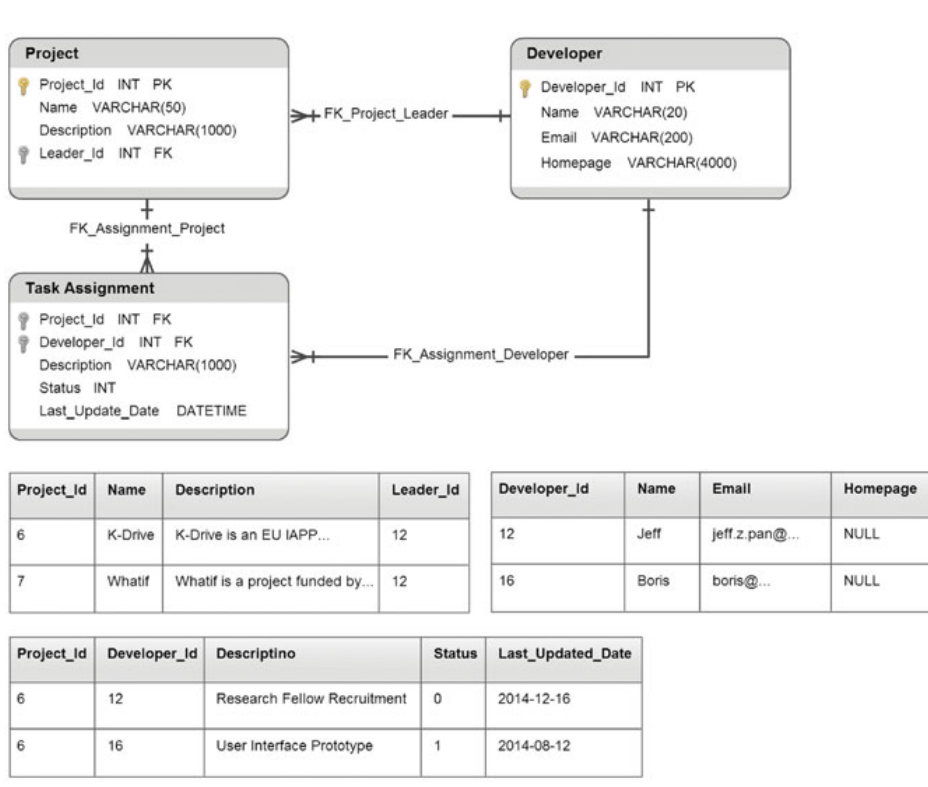
There are various approaches available to perform the data lifting. For example, to extract knowledge from natural language texts or Web pages, there are approaches of named entity recognition, information extraction, concept mining, text mining, etc. There are many tools or libraries available in either open source or commercial licenses, such as GATE,13 OpenNLP14 and RapidMine.15. This is for **Web-Search** Data.

In this section, we lay special focus on two W3C standards on data lifting, i.e. RDB2RDF and GRDDL. These two standards cover the data lifting from structured or semi-structured legacy data and are probably the most important data formats in large organisations. RDB2RDF specifies how to translate relational data into the RDF format (introduced in Sect. 2.3.1) and GRDDL defines the standard approaches to translate the XML data into RDF (briefly presented in Sect. 2.3.2). These are for **Enterprise KGs**



**RDB2RDF**

The first recommendation is called “A **Direct Mapping** of Relational Data to RDF”.16 If you prefer a quick conversion and your database schema is designed to be good enough (e.g. well-defined primary keys and foreign keys, meaningful table and column names, etc.), then direct mapping can be a good choice. The only input in this case is the database (data and schema) and the output is the RDF version of your data. It is simple but you don’t have much control over the conversion settings. The second recommendation is “**R2RML**: RDB to RDF Mapping Language”.17 Using an R2RML, you can customize the mappings to generate the RDF data based on your design. For example, if you would like to generate your RDF data by reusing some popular vocabularies or your predefined domain ontologies, you will go for R2RML. Table 2.2 gives some of the possible considerations for making a choice between the two specifications. See example on P56!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



Customized conversion (RDB2RDF):

Term Maps

Logic Table (with SQL)

Triple Maps

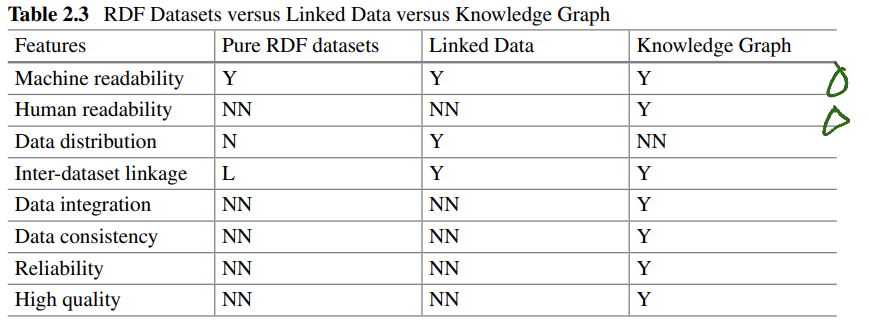
<http://www.w3.org/TR/rdb2rdf-implementations/>

**GRDDL**

In addition to residing in relational databases, the legacy data to lift might also be stored in other formats. Among others, a common one could be the **XML** format. For example, the purchase orders from the retailer section are encoded in an XML format, the data captured by sensors uses XML syntax or the information in question is simply published as Web pages (XHTML) on the Web. Although all these data are stored in XML, their syntaxes and semantics are potentially totally different, which impedes their integration to the organisation’s knowledge graph.

* 1. **KG & Linked data**

A knowledge graph is a structured dataset that is compatible with the RDF data model and has an (OWL) ontology as its schema.



* 1. **For Web search and for Enterprise**

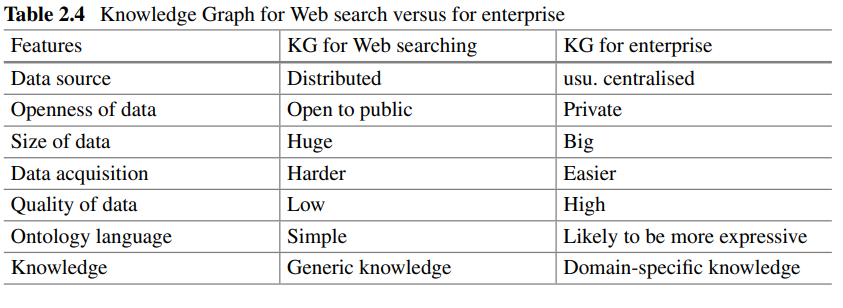
**For Web searching**

The Google Knowledge Graph provides a short summary about the topic with structured information, as well as a list of commonly used links to the other sites in order to back up for the most possible queries on that topic.

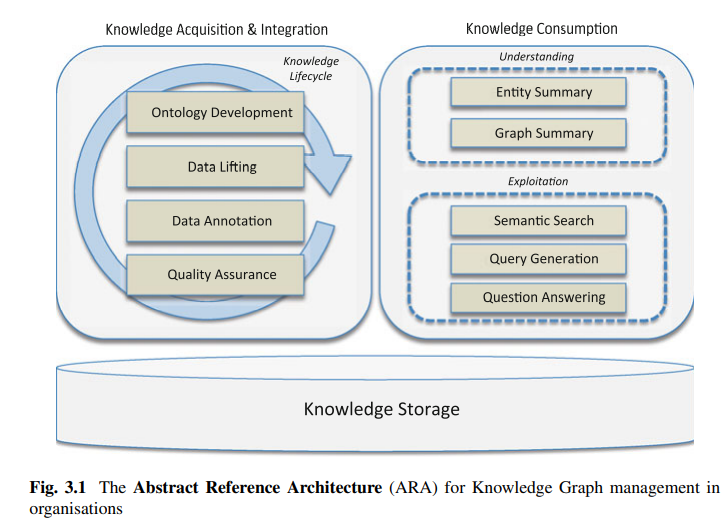
**For enterprise KG**

This is generally achieved by creating data infrastructures, developing applications working on them, managing processes for data acquisition, curation, maintenance, integration and access.

Primarily, organisations may provide unified knowledge schemas (ontologies) for their business and spread them along corporate branches. These ontologies can be built on the basis of existing business vocabularies and industry models, and are easily workable to follow the business evolution



1. **Knowledge Architecture for Organizations**
   1. **Overview**



However, we will first discuss how to avoid duplicating the storage of data you already store in some other format (e.g. as relational databases) by adding a translation layer in order to access those data-sources using the conceptual layer defined by your knowledge graph.

* 1. **Acquisition and Integration (Ontology, NER, Ontology learning)**

**Ontology development**

With lightweight schemas you only name the main relations between the entities in your data (you don’t even need to name the types of entities). A more heavyweight schema will contain formal definitions with restrictions on what specific entity types mean in your context.

The importance of applying engineering principles to the ontology development process has been recognised for a long time now and several methodologies have been proposed for this purpose including METHONTOLOGY [75], Diligent [248], HCOME [138], NeOn [226] and DOGMA [150]. Typically, such a methodology defines a set of activities that need to be performed while developing an ontology and usually suggests or provides methods and techniques for effectively carrying out these activities’ tasks.

For non-ontological resources. You can define the schema for your knowledge graph from scratch by analysing your use cases or your domain; this is the top-down approach. However, you can also try to reuse schemas about existing databases or classification schemas in your organisation (or existing public schemas). The main problem here is that you need to translate these schemas into RDF or OWL. This process can typically be mostly automatised because the existing schemas and classifications are lightweight, and thus can be captured faithfully by RDF or OWL.

**Text – NER – Thematic (Central Role)**

Integrating texts into knowledge graphs is typically done by means of two tasks, namely Named Entity Resolution and Thematic Scope Resolution. The first task involves detecting mentions of named entities (e.g. people, organisations or locations) within texts and mapping them to their corresponding entities in the knowledge graph source. On the other hand, the Thematic Scope of a document can be defined as the set of semantic entities the document actually talks about.

P78

* 1. **Storing and Accessing Layer (RDF stores, Property graph-based)**

This is where graph databases and triple stores fill the gap, because they are flexible regarding the schema: they can be used with no schema or with a lightweight schema, or they can be used with various schemas.

**Property Graph-based**

RDF Stores are not the only way to store knowledge graphs. There are alternative data models similar to RDF which can also be used. In this section, we briefly introduce property graphs, an alternative data model (not based on RDF) for representing knowledge graphs.

Storage of Property:

The features of property graphs discussed above suggest that, like RDF datasets, property graphs are very flexible data models and can be used to represent many other existing data models, which means that they can be stored in RDF triple stores, relational databases, NoSQL databases, etcP90

The **Schema**

DF and Property Graph, by contrast, can be schema-less or schema-free. Extending the amount of data related to an element can easily be done without needing to adapt any schema. Any other nodes or edges in the graph will not be affected. RDF, in terms of flexibility, is even better than the Property Graph, because it also allows simple schema (with RDFS) or rich schema (with OWL).

* 1. **Consumption layer (Semantic search, Query Generation and Answering)**

In this section, we will introduce some of these consumption paradigms, such as semantic search, automatic summarisations and question answering. Although we do not explicitly mention reasoning here, reasoning is in fact required for supporting many of these tasks.

**Semantic Search**

During indexing, the knowledge graph can be used to automatically generate a list of entities relevant to the organisation and their synonyms.

Entity summary

Graph summaries & Graph profiling & Graph analytics

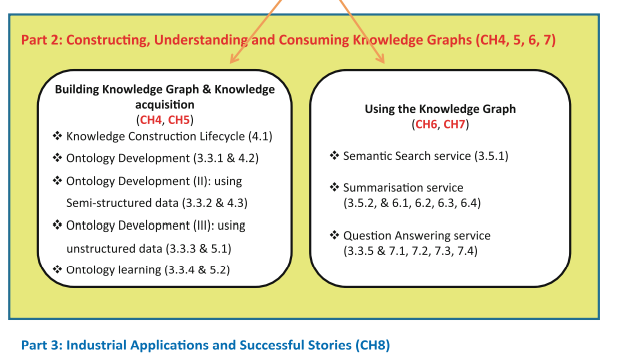
**Query Generation**

In terms of query generation for knowledge graphs, it is more interesting and important to unlock the ability of revealing interesting information in the graph to users.

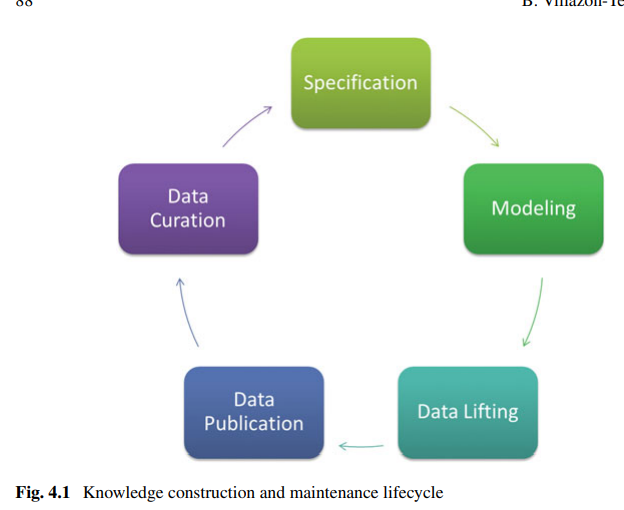
**QA**

Historically, question answering has been attempted on top of relational databases. However, since relational databases have a fixed schema, have a specific purpose (to serve as the backend of a particular application) and do not contain additional

**Part II Constructing, Understanding and Consuming Knowledge Graphs**



1. **Construction of Enterprise KG (I)**
   1. **Construction and maintenance lifecycle**



1. **Specification**

(1) identification and analysis of the data sources, and

(2) URI design.

Use meaningful URIs, instead of opaque URIs, Use slash (303) URIs, instead of hash URIs, Separate the TBox (ontology model) from the ABox (instances) URIs

1. **Modeling**

The most important recommendation in this context is to reuse as much as possible the available vocabularies. This reuse-based approach speeds up the ontology development, saving time, effort and resources. IF not, using NeOn.

1. **Data lifting (ImportantP106)**

Transformation and Linking

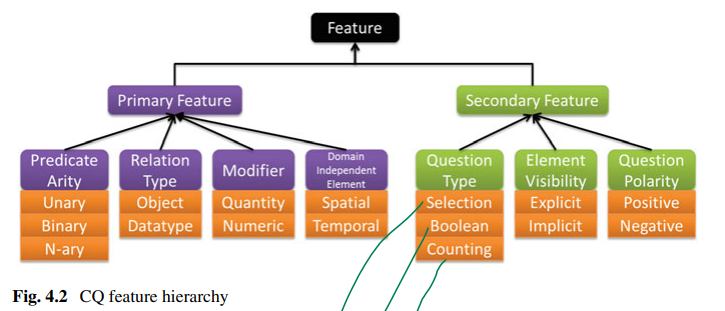
1. **Data publication**
2. **Data Curation**
   1. **Ontology Authoring (Question-Driven approach)**

In this section we introduce the methodology of Competency Question-driven Ontology Authoring (CQOA) [200],35 which leverages the ideas of competency questions and testing driven software development (where a suite of tests represent a specification for a program and the tests are coded against).

**Competency Questions**

Compared to more formal requirement specifications, CQs are particularly useful to ontology authors who are less familiar with description logics because CQs are in natural language, are about domain knowledge and do not require an understanding of DLs. Hence in ontology authoring practice, CQs help authors to determine the scope and granularity of the ontology, and to identify the most important classes, properties and their relations.

Formulation of Competency Questions



**P115 CQ** Archetypes

**P118** Authoring Tests

* 1. **Semi-automated Linking of enterprise data for virtual KG (Semantic tagging and data linking)**

After having done a good knowledge modelling job, the next step in constructing a knowledge graph naturally shifts to the data level. According to the lifecycle mentioned in Sect. 3.5, this should be the data lifting step. In this section, we introduce an approach for creating data linkage that is a critical type of knowledge in knowledge graphs. Specifically, we describe Helix, a system for creating links among large-scale and heterogeneous information sources in large organisations.

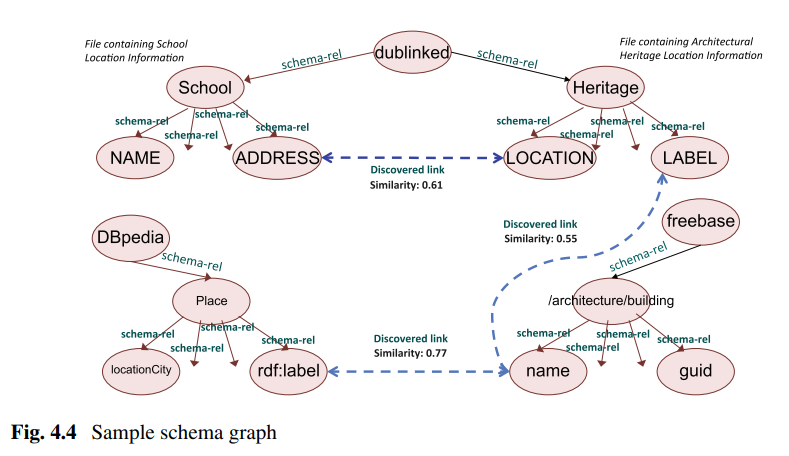
One key point we make is that there are many possible classification techniques—we articulate one possible mechanism to automatically tag datasets, but this is doubtlessly an area for fruitful research, and we need many more techniques in this space targeting data discovery and data classification.

To summarise, the techniques that relate to the problem of data discovery include methods to (a) normalise data in different formats, (b) index structured data in tables, (c) perform semantic matching between schema elements of structured data, (d) tag data with semantic tags and (e) find linkage points in the data so that users can join between tables. In this chapter, we focus on techniques (c)–(e) and describe how one might use these components in a larger system for guided data exploration, and show use cases from actual scenarios.

One of the goals in Helix is to process data based on explicit user needs and avoid unnecessary or expensive pre-processing given that we are dealing with largescale enterprise data. Therefore, the data pre-processing phase comprises only three essential steps, all performed in a highly scalable fashion implemented in the Hadoop ecosystem: (a) schema discovery, where each input source schema is represented in a common normalised model in the form of a local schema graph; (b) full-text indexing, where data values and source metadata are indexed; and (c) linkage discovery that incorporates instance-based matching and clustering of the (discovered) schemas; (d) semantic tagging and schema linking, the outcome of the pre-processing phase is a semantically tagged Global Schema Graph; (e) linkage point discovery, the use of linkage point discovery to find possible points for fuzzy joins. In the following, we discuss briefly steps (c)–(e).

**Linkage Discovery** (Relate to Hadoop)

We address this problem by casting it into the problem of computing document similarity in Information Retrieval [70]. Locality Sensitive Hashing (LSH) techniques. Briefly, we construct a fixed small number of signature values per attribute, based on MinHash [41] or Random Hyperplane [47], in a way that a high similarity between the set of signatures guarantees high similarity among instance values. P123



Scenario1 & 2

Scenario 3 Customer relation Management in Large Organizations

In Conclusion:

In this section, we described Helix, a system that allows knowledge workers and data scientists to explore a large number of data sources using a unified intuitive user interface. Users can find portions of the data that are of interest to them using simple keyword search, navigate to other relevant portions of the data and iteratively build customised views over one or more data sources. These features rely on highly scalable schema and linkage discovery performed as a pre-processing step, combined with online (and in part social) guidance on linkage and navigation. We demonstrated capabilities of our system through a number of usage scenarios

**5. Construction of Enterprise Knowledge Graphs (II)\***

In this chapter, we continue with the Acquisition and Integration Layer of Chap. 3’s reference architecture, focusing on knowledge graph construction techniques. Nevertheless, we shift from semi-automated approaches to automated approaches of knowledge graph construction by describing two additional frameworks, one for entity/scope resolution of textual data (Sect. 5.1) and one for the learning of ontological schemas from data (Sect. 5.2)

**5.1 Named Entity, Thematic scope**

In this section, we describe Knowledge Tagger (KT), a framework that performs Named Entity and Thematic Scope Resolution in texts using relevant domain ontologies and semantic data as background knowledge. Its distinguishing characteristic is its disambiguation-related customisation capabilities as it allows users to define and apply custom disambiguation evidence models, based on their knowledge about the domain(s) and expected content of the texts to be analysed.

Entity Reference Resolution Process (Algorithm)

Thematic Scope Resolution…

Framework application evaluation

**5.2 Schema learning for KG**

In this section, we describe a way to deal with the schema learning problem **from incomplete** Web data. As we know, **ontology TBoxes, or conceptual schemas**, are the backbone of Knowledge Graphs, but they are always difficult to obtain, especially when the data is incomplete. In our approach, the TBox learning task in a Description Logic (DL) is transformed into a Bayesian inference task in an extension of the Bayesian Network, which is based on the original DL ontology. **Bayesian Description Logic Network (abbreviated as BelNet),** integrating the probabilistic inference capability of Bayesian Networks with the logical formalism of DL ontologies, supports promising inference, even when the dataset is incomplete. In this section, we first introduce the motivation for this work, explain the details of BelNet+ and, finally, introduce a TBox learning approach with BelNet+ based on Open World Assumption (OWA). In order to showcase the performance of this approach, a novel evaluation framework with incomplete data will be adopted to conform to the open, dynamic and non-consistent Web environment. Finally, the result from empirical studies on comparisons with the state-of-the-art TBox learners is provided, verifying the effectiveness of our approach.

**Motivation:**

The knowledge acquisition bottleneck has resulted in inexpressive schemata (also known as TBoxes, while the data part of ontologies iscalled ABoxes) on the Semantic Web [52, 125]

P143 In an environment like the Semantic Web, data generally suffers from incompleteness [263], which consequently hinders the learners from getting correct results. In this section, we **focus on learning the TBox and from incomplete ABox data.**

In order to address the above challenge we make the following four contributions in this section.

• We **generate the negative examples** according to the CWA in a manner similar to that mentioned in [140]. However, to solve the noise issue brought by CWA and incompleteness, we adopt an approach that instead of merely considering the instances of concept pairs, uses also inference in a Bayesian network that leverages the structure in the data.

• In order to foster promising inference on subsumption and disjointness axioms, we extend BelNet [263] to BelNet+. BelNet combines Bayesian networks with DLs by representing DL concepts as nodes and subsumptions with links. In BelNet+, we extend the semantics of links in BelNetly **using additional links for disjointness**. Compared to BelNet, BelNet+ is more effective in detecting disjoint concepts and answering queries.

• We consider the TBox learning as instance classification. In order to conform to the Open World Assumption (OWA) generally made in the Semantic Web, we **extend the traditional confusion matrix by considering unknown results** (neither true nor false), and propose the metrics using the new confusion matrix correspondingly. Our extension of traditional evaluation metrics reflects more objectively on the performance of the learners.

• In order to evaluate the state-of-the-art TBox learners, we set up gold standard ontologies correspondingly. Meanwhile, in our evaluation framework, the quality of the gold standard ontologies is more easily guaranteed.

In the rest, we first introduce the BelNet+ model and the TBox learning with it. Second, we describe an evaluation framework for TBox systems and show empirical performance evaluations. We also briefly review the related works and discuss the future research

**BelNet+**

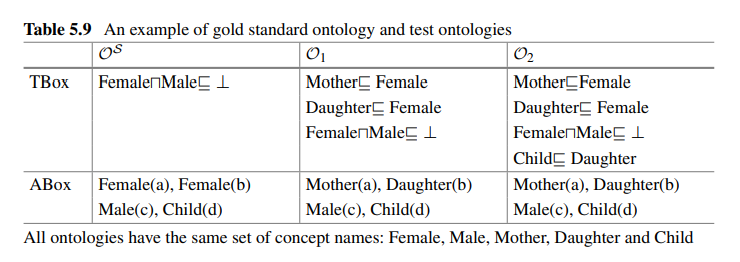
The syntax, the Semantics, Parameter Estimation

Structure Learning (Structure Score, Structure Search)

**Terminologies (TBox) and assertions (ABox)**

A novel evaluation framework

In order to incorporate the unknown values in the classification results, we extend the traditional confusion matrix used in the evaluation of binary classification [108] by considering “unknown” as a specific classification result (cf. Table 5.8).



**Experiments**

We have implemented a prototype of BelNet+ with the TBox learning algorithm in Java and Scala. We designed and carried out the experiments to highlight the effect of incompleteness on learning methods. In this section, we evaluate the performance of the proposed learning method focusing on answering the following three questions: (1) How promising is the inference in BelNet+? (2) How are the performances of the four approaches, namely DLLearner, GoldMiner, BelNet and BelNet+, under the existence of incompleteness? (3) Will the amount of incompleteness be decreased with TBox learning?

**Related Work and Summary**

Inductive Logic Programming. Inductive logic programming (ILP) marries Machine Learning and data mining, whose survey can be found in [62, 73].

Association Rule Mining. As a classical data mining method for mining relationships, association rule mining (ARM) is applied in TBox learning problem.

Statistical Relational Learning. Koller et al. extended DL CLASSIC with nodes in a BN representing probabilistic information of the individuals in a specific class [137], and the model was called P-CLASSIC.

What is the entity description pattern?

The main novelty of our summarisation approach is that it summarises an RDF graph by another much smaller graph structure based on atomic graph patterns.

**6. Understanding Knowledge Graphs**

The enabler of this feature comes from the fact that data semantics are explicitly represented in knowledge graphs instead of being in the business logic layers of applications. But, to realise such feature, the questions are how such explicit data semantics can be utilised to serve the end users? and what kinds of applications knowledge graphs can directly support? From both academia and industry, many efforts have been put to answer these questions.

• Understand Entities Knowledge graphs are essentially graphs of interlinked entities. The most straightforward application is in **helping users understand entities in such graphs**. For example, if you search UK in Google, Google’s knowledge graph can directly present you the key facts of the United Kingdom like abstract, population and dialling code, and also key related entities of the UK, such as its capital city and main destinations. Section 6.1 discusses the challenges of entity understanding in knowledge graphs with open data and introduces a concept space-based approach for summarising individual entities.

• Exploit Knowledge Graphs While entity understanding is exciting and very efficient for retrieving common knowledge of entities, sometimes people are more interested in questions for which the answers are not directly available. For example, what are the five coldest years in the UK in the last 50 years. Although such a fact is not directly available (as an attribute of UK), the knowledge graph might have enough data to draw the answer of the question (e.g. through logical based reasoning). In such cases, efficient exploitation techniques might help the user locate the right portion of the knowledge graph and provide them the right facilities to draw the answers by themselves. Section 6.2 introduces **a knowledge graph exploitation system that identifies entity description patterns as building blocks for guiding users** in their knowledge exploitation tasks.

• Profile Knowledge Graphs Data scientists in large organisations might need to make use of multiple knowledge graphs in their daily work, e.g. linking various knowledge graphs to derive knowledge for decision-making. Choosing the right knowledge graphs is the very first step to conduct such tasks. This requires an effective way to determine whether their tasks can be solved by certain knowledge graphs. Knowledge graph profiling is a typical application for serving enterprise users. Section 6.3 presents a knowledge graph profiling approach for users to assess **whether knowledge graphs can meet their goals**.

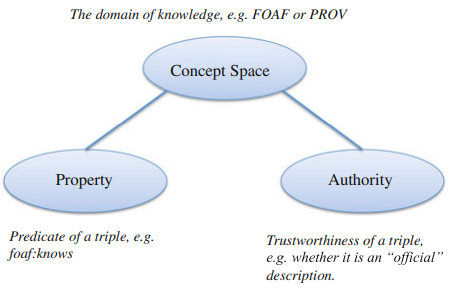
• Reveal the Insights from Knowledge Graphs Querying knowledge graphs might be one of the most effective and efficient ways to retrieve relevant knowledge from them. However, for end users, the task of **constructing correct structural queries** itself is already beyond efficient. Automatically generating queries for knowledge graphs or even revealing insights directly from them can be extremely useful techniques for end users to make use of knowledge graphs. Section 6.4 introduces an automatic query generation approach for identifying insights into knowledge graphs

**6.1 The Things in KG**

Using URIs to identify entities makes data integration for the same entities much easier across different knowledge graphs or different portions of the same knowledge graph. This is one of the best advantages to support data integration using Linked Data techniques。 However, **the challenges are: unordered nature of the RDF, data redundancies, data qualities, huge-data-volume**)

In this section, we briefly introduce an approach called Concept Space based Summarisation [253], which targets to deal with three of the above four challenges, i.e. numbers 1, 3 and 4. For the challenges of orderless and data quality issues, it proposed a three-dimension organisation, which organises the data in dimensions of concept spaces, RDF triple predicates and data authority. For the huge data volume challenge, it proposed a summarisation technique that assesses the importance of RDF data and extracts the important subset for achieving a quick and comprehensive understanding of entities.

**Two challenges to automatic clustering** or classification generation: (1) how many clusters or classes are appropriate? and (2) how to generate meaningful clusters/classes. To tackle the two challenges in an RDF data organisation, the Falcons search engine proposed a hierarchical classification approach (cf. Fig. 6.1) to classify the information space about Semantic Web entities, a.k.a. organising RDF triples that describe entities.



In the first level of classification, RDF triples are classified by a notion called concept spaces. The idea of this level of classification is to group data by domains.

**Concept Space Classification**

Approaches (One is to use hierarchical classification method, another is to cluster individual concepts. The three is a combination of the two)

One very useful community contribution of the vocabulary classification comes from the LOV (Live One Vision Project) project,9 where 475 popular Linked Open Data vocabularies are grouped into a hierarchical classification with the first level having 11 groups.

P171

**Property Facet**

Orderless Challenge: RDF 🡪 HTML

Authority Facet

**Summarisation of Entity Data**

In the Falcons system, an importance function Imp (cf. Formula 6.1) is used to calculate the importance of an RDF triple. P172

**6.2 Entity description**

So far, Linked Data principles and practices are being adopted by an increasing number of data providers, getting as a result a global data space on the Web containing hundreds of LOD datasets [32]. However, the technical prerequisites of using Semantic Web datasets prevent **efficient exploitation** on these datasets. To tackle this problem, the Linked Data community has been putting a lot of effort into it. We classify such efforts into two groups: (1) work that deals with the **extraction of metadata** from the datasets and (2) work related to **dataset summarization.**

Extracts metadata from datasets: P174 LODStats, make-void…

The rest of this subsection is organised as follows: First, we introduce the details of the summarisation definition and generation. Then, we illustrate three exploitation tasks of (**Quick Understanding**) big picture presenting and summary browsing, (**Guided Exploitation**) two query generation methods and **(Dataset Enrichment**) atomic pattern-based dataset linkage. Finally, we discuss the graph pattern summarisation techniques and future directions. The rationale behind the assumption is that RDF data exploitation is usually based on graph patterns, e.g. SPARQL queries are based on the basic graph patterns (BGP).

Specifically, in this section, we propose one definition of such building blocks, i.e. **Entity Description Patterns (EDPs for short)**, which is stated in Definition 1.

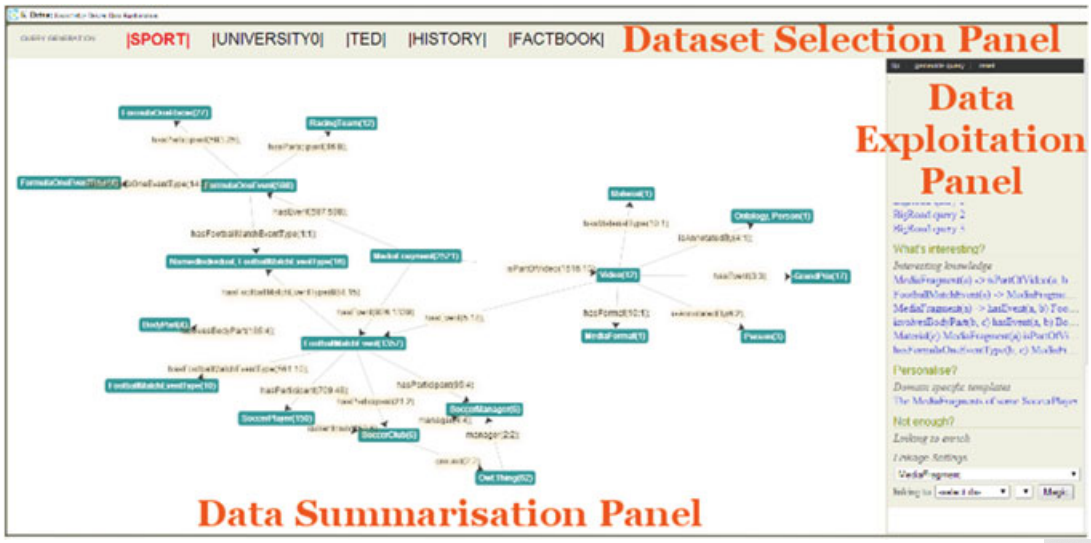
Definition 1 (Entity Description Pattern)

Definition 2 (EDP Merge)

Definition 3 (EDP of RDF Graph)

Definition 4 (EDP Graph)

(Arbor Javascript Library (http://arborjs.org/introduction) is used for the EDP graph rendering)



Task1: **The Big Picture**

In data summarization panel: Summarization graph + node browsing (description) + graph browsing (select a node and enter a subgraph)

Task2: **Query Generation**

Based on the EDP summarisation, we implemented two types of query generation techniques. One is called the guided query generation, which generates queries by utilising the EDP graph and statistics information attached in the graph. The other generation technique (which we will discuss in detail in Sect. 6.4) makes use of the links in the summarisation to perform efficient association rule mining.

Task3: **Enrichment**

Link to another KG

**6.3 Profiling KG: Summarization**

In this section, we are interested in facilitating the generation of requirement-oriented and task-specific KG summaries that may help knowledge engineers and data practitioners assess whether and to what extent a given KG is suitable for the task at hand.

For: (i) explicitly express the requirements that a KG needs to satisfy for a given **task or goal** and (ii) automatically **measure/assess** the extent to which a KG satisfies each of these requirements and compile a summary report. To implement these two capabilities, we follow a checklist-based approach.

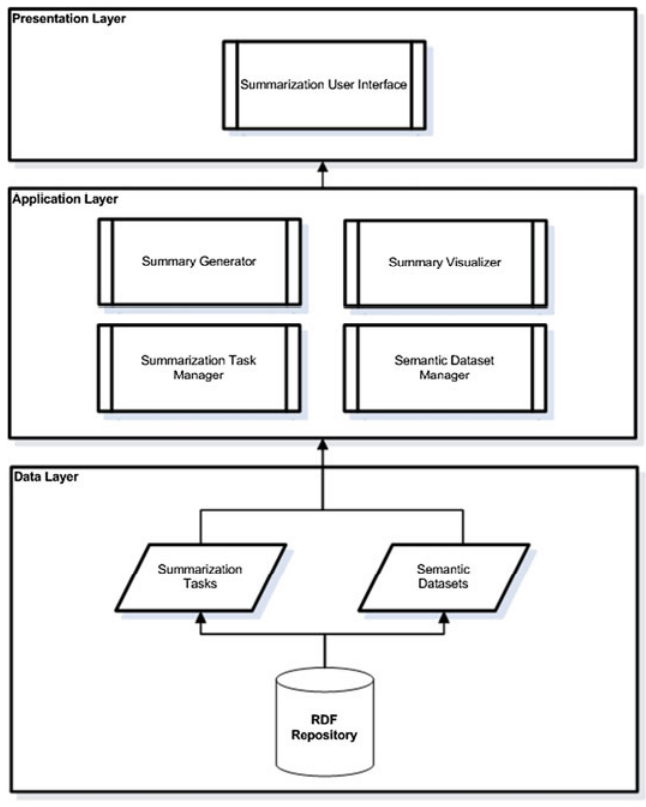
Summarisation Task Representation:

Goals, Requirements, Data Analysis Operations

Summarisation Task Creation:

To create a summarisation task one needs to define its goal(s), its requirements and the association to these operations. Some high-level requirements that we have already identified and may be used for multiple goals are the following: Coverage, labelling adequacy and richness, Connectivity…

Implementation(P182)



Application Example (The company wants to make these images easier to find using a semantic search over the textual descriptions)

**6.4 Query Generation (P187)**

In this section, we propose a tractable query generation approach based on data summarisation and graph patterns

Definition 5 (Graph) A labelled, directed multiple graph

Definition 6 (Graph Operations) A graph N1, E1, M1, L1 is a subgraph of another graph

Definition 7 (Graph Pattern) A graph pattern is a graph in which some nodes are variables.

Definition 8 (Instance) A substitution S = (v1 → v2) replaces a vector of variables v1 with a vector of URI references/literals/blank nodes/variables v2.

Definition 9 (Looped Graph Pattern) A graph pattern is a looped graph pattern if it contains a circle of nodes.

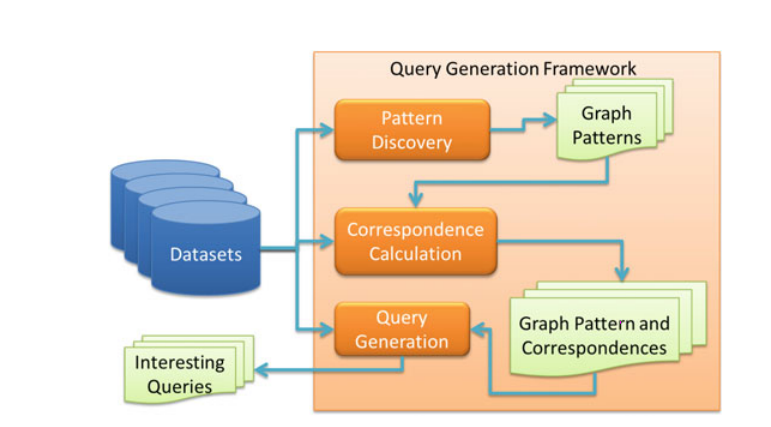
Proposition 1 (Query Answering) Given a dataset D, for any conjunctive query BGP Q of a vector of variables v1, let GP be its corresponding graph pattern. (P190)

Definition 10 (Graph Pattern Correspondence)

Theorem 1 Two graph patterns GP1 and GP2 correspond on variables v IFF Q(GP1) ∧ Q(GP2) has a solution:

1. Queries with strong Correspondence
2. Queries on Exceptions

**Query Generation Framework**



we examine three different approaches (worst-case polynomial time) **to find the corresponding graph patterns.** FOIL (First-Order Inductive Learning) constructs the graph pattern by including a set of possible reachable variables via gradually growing the graph pattern. The algorithm selects the best variable from the set by a gain function (cf. [196]). FOIL tends to generate star-shaped graph patterns. COMB and LOOP approaches utilise the association rule mining technology. They tend to generate chain-shaped or looped graph patterns.

**Evaluation of the Query Generation Method (An example)**

**Conclusion and Future Work**

This section presented a novel and tractable approach to generate candidate insightful queries for knowledge graphs. A combination of data summarisation and different mining technologies has been exploited to extract graph patterns and construct candidate insightful queries. The evaluation shows that the proposed framework can generate insightful queries from synthetic and real-world datasets. In addition to QG based on systematic parameterisation [92], it might be worth exploring how to extend the described framework in order to embed the support of tractable reasoning and schema [186], so that, on the one hand, reasoning can be done on the fly with data summarisation and on the other hand, reasoning can be used to eliminate query pairs that are inferred to share all (or no) solutions.

1. **QA in KG**

Knowledge graphs play a key role in question answering. On the one hand, they are the natural encoding for structured knowledge extracted from texts, databases or other sources, making it available for efficient queries.

**7.1 Over Text Doc.**

This section provides a brief introduction to Question Answering (QA) systems over natural language text documents. It gives a general discussion about the technical approaches that are usually used in implementing a QA system. Two of these approaches, are further elaborated in this part, which are Natural Language Processing (NLP) and Answer Ranking, respectively, are further elaborated in this part.

**Relalising a QA System**

The QA systems have applications in different scenarios. They can be used to efficiently find information on the Web or in specific document collections. Since they can be tailored to a given domain there is interest in their usage in medical and legal fields or in specialised knowledge bases owned by companies, agencies, public administrations and research centres. Moreover, they are very effective in finding information when every document in the collection is equally important and the hyperlinks between them can be neglected. Modern QA systems employ methods and theories coming from different fields: Information Retrieval, Natural Language Processing and Machine Learning.

**The NLP Step**

NLP Pipelines are used to wrap, compose and orchestrate the different tasks and their inputs and outputs, and facilitate their reuse. A pipeline is a sequence of components performing different NLP tasks

UIMA (Unstructured Information Management Applications) is a framework for building systems composed of components whose interfaces are defined in terms of input and output.

**The typical NLP pipelines**

Tokenisation 🡪 Stemming 🡪 Lemmatisation 🡪 Sentence Boundary Disambiguation 🡪 Named Entity Recognition (Can be domain-specific) 🡪 Part-of-Speech Tagging (POS) 🡪 Chunking 🡪 Syntactic Parsing (Keep the most probable parse tree) 🡪 Relation Extraction (RE) 🡪 Semantic Role Labeling (SRL) 🡪 Co-reference Resolution (Discourse Parsing)

**The Ranking Step**

The purpose of ranking is to order a list of objects using their features with the goal of putting the more relevant objects first in the list. The relevancy of an object can be subjective and depends on the problem at hand and the available data.

**Learning to Rank:**

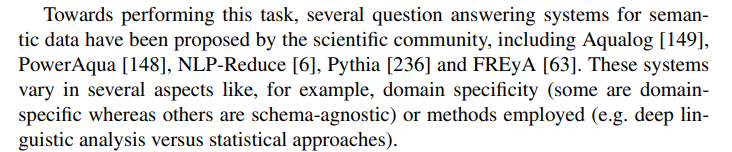
The methods using Machine Learning to perform ranking are called Learning to rank (LTR) methods. The general process performed by learning to rank methods concerns collecting training data, such as queries and associated labelled documents, performing feature extraction on them, learning a ranking model by minimising a loss function on the training data and using the learned model to infer the ranking of new data. The ranking model in learning to rank methods is feature based. Most of the state-of-the-art algorithms learn the best way to combine features extracted from query and document pairs using **discriminative training** [147]. (P203) Thus, the ranking problem is approximated by a regression problem.

**Evaluation of Ranking**(P204):

Precision at Position k; Average Precision; mean average precision (MAP); mean reciprocal rank (MRR);

**7.2 QA over KG**

The main task to be tackled is the transformation of the natural language query into a set of required RDF2 triples, typically expressed in the SPARQL3 query language and in accordance with the system’s ontology.



**State-of-the-Art Approaches (**Relevant approaches in that category include FREyA [63] and PowerAqua [148].) (P206)

In particular, the goal of FREyA is to develop user-friendly interfaces to Linked Open data. They emphasise the use of clarification dialogues with the users when it encounters as-then unresolvable ambiguities from which it trains itself further for future queries.

On the other hand, the focus of PowerAqua is the integration of information from multiple, heterogeneous semantic resources to derive answers to natural language queries. This requires, first, efficient and accurate identification of which semantic resources are even useful for answering the query. Further, due to the inevitably extensive volume of data potentially being handled, they consider scalability issues, and the issues inherent in third-party sources of noisy and incomplete data.

On the other hand, AquaLog [149], a precursor of PowerAqua, does not rely on a priori manual linguistic enrichment of the ontology. (P208)

Some important **differences between** enterprise knowledge graphs and non-enterprise:

1, Higher quality

2, Less heterogeneous

3, More predictable in content

4, Answering questions reliably and consistently is more important

5, Transparency and explicability

6, Should be improved by feedback

**QA systems implement pipeline**

• Stage 1: Question Linguistic Analysis

• Stage 2: Entity Mapping and Disambiguation

• Stage 3: Formal Query Construction.

• Stage 4: Query Execution and Answer Provision.

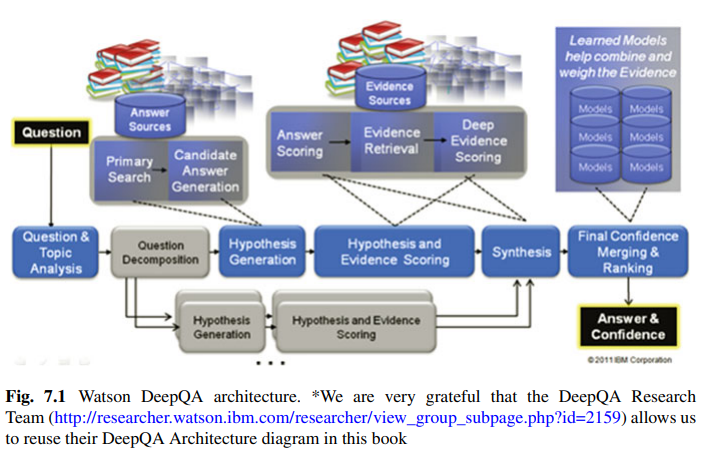
For the first two stages (the hardest):

1, Restricted subset of natural languages, like GiNSENG

2, Template-based approaches, such as LODQA. Construct a query template on the basis language

3, A joint manner

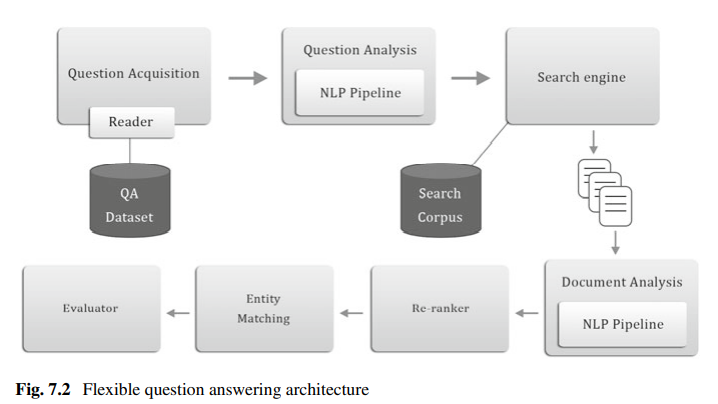
**7.3 Waston DeepQA**



Specifically, knowledge graphs were used for temporal and spatial reasoning, detecting and scoring temporal/spatial relations and computing the YAGO type coercion (TyCor). Here, a brief description of the two processes is given. Interested readers can refer to [127] for detailed discussions.

**7.4 QA system that combines text analysis and KG**

Specifically, this section starts with a brief introduction of a NLP pipeline well suited for working with knowledge graphs. Then, it will put special emphasis on the re-ranking module. The re-ranker is based on the structural kernel technology, which enables the easy encoding of trees and graphs in learning algorithms, such as SVMs. This way, it could exploit two different types of graph information: (i) syntactic and semantic graphs derived by the syntactic and semantic analysis of the question and answer passage pairs; and (ii) external Knowledge Graphs such as DBpedia, which we exploit for enriching the previous graph with semantic and typed information.



The steps executed by the systems are the following:

1. the questions and the data for evaluation are read from the dataset; in the interactive version, a question is read from the user input;

2. the question is analysed by the NLP pipeline;

3. the question is used to query the search engine and retrieve the top-k relevant documents;

4. the documents are analysed by the NLP pipeline;

5. the question classifier output and the named entity recognition tasks’ (NERs’) output are used to reorder the retrieved document list;

6. the re-ranker is used to produce a permutation of the retrieved document list;

7. the evaluation of the document list is carried out and the results are averaged for all the questions. In the interactive version, the entity matching the question type is returned

**Exploiting External Knowledge (Graphs) for Re-ranking(P220)**

Shallow Chunk Tree (CH)

Using Wikipedia for REL Matching (wiki)

CH + Question Classification + Focus Detection (CH+QC+TFC)…

SVM Classifier

**Evaluation: Impact of Knowledge Graphs in Semantic Structures**

Dataset

Search engines.

Wikipedia link annotators

**Part III Industrial Applications and Successful Stories**

**8 Success Stories**

**8.1 IN the Media Industry**

**8.2 In Cultural Heritage**

**8.3 In healthcare**

**9 The future**