Semantic Web 0 (0) 1
IOS Press

Data-driven Methodology for Knowledge Graph Generation within the Tourism Domain

Alessandro Chessa [a], Gianni Fenu [b], Enrico Motta [c], Francesco Osborne [c,d],
Diego Reforgiato Recupero [b], Angelo Salatino [c] and Luca Secchi [a,b,*]
[a] *Linkalab, Viale Elmas, 142, 09122, Cagliari, Italy*
E-mail: alessandro.chessa@linkalab.it
[b] *Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy*
E-mails: fenu@unica.it, diego.reforgiato@unica.it, luca.secchi@unica.it
[c] *Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom*
E-mails: enrico.motta@open.ac.uk, francesco.osborne@open.ac.uk, angelo.salatino@open.ac.uk
[d] *University of Milano Bicocca, Milan, Italy*

Abstract. The tourism and hospitality sectors have become increasingly important in the last few years and the companies operating in this field are constantly challenged with providing new innovative services. At the same time, (big-) data has become the "new oil" of this century and Knowledge Graphs are emerging as the most natural way to collect, refine, and structure this heterogeneous information. In this paper, we present a methodology for semi-automatic generating a Tourism Knowledge Graph (TKG), which can be used for supporting a variety of intelligent services in this space, and a new ontology for modelling this domain, the Tourism Analytics Ontology (TAO). Our approach processes and integrates data from Booking.com, AirBnB, DBpedia, and GeoNames. Due to its modular structure, it can be easily extended to include new data sources or to apply new enrichment and refinement functions. We report a comprehensive evaluation of the functional, logical, and structural dimensions of TKG and TAO.

Keywords: Knowledge Graphs, Ontology Design, Tourism Ontology, Web Science, Web Mining, Tourism, Hospitality

1. Introduction

We are currently living in the age of big data, and the sheer volume of new data being generated is making the World Wide Web shifting from a web of content to a web of data. This gives all practitioners the opportunity to build more innovative and functional web services.

Semantic Web and Linked Data technologies aim to represent the web itself through a large global graph that can be queried using standard protocols and languages [39]. The World Wide Web Consortium (W3C) has developed and promoted different standards, like RDF/S, OWL and SPARQL, that are now widely adopted to create knowledge bases that represent data as knowledge graphs (KGs). A knowledge graph is a graph of data whose nodes represent entities of interest and whose edges represent relations between these entities [23]. A few examples of knowledge graphs publicly available are DBpedia [39], YAGO (Yet Another Great Ontology) [55] or WikiData [22]. Knowledge graphs can store data and metadata using a common structure and are often used in application scenarios that involve extracting and integrating information from multiple, and possibly heterogeneous, sources. Typically the data in the

[*]Corresponding author. E-mail: luca.secchi@unica.it.

knowledge graph are modelled according to a domain ontology, which gives meaning to the represented information and supports inferring new knowledge.

The field of tourism is a natural domain of application of these technologies since stakeholders in this space need to integrate data from several heterogeneous sources in order to generate a multifaceted characterisation of tourist destinations and all relevant actors [7, 26, 58].

A tourist destination can be thought of as the place or area which is central in the decision of a tourist to take the trip[1] and is usually characterised according to two aspects: supply and demand. The supply side is based on the willingness and ability of producers to create goods and services to take them to market. Understanding the supply side of tourism includes all aspects related to tourism offerings and attractions (e.g., accommodations, events, points of interest, restaurants, and so forth). On the other hand, demand refers to how much (quantity) of a product or service is desired by buyers. Understanding which factors influence the demand side of tourism includes all aspects related to tourists' choices and opinions or their characteristics (e.g., socio-demographic, classification, provenance).

This information is crucial for informing business and marketing decisions as well as supporting a variety of software and services in this space, such as search engines and recommendation systems [37, 57].

The creation of KGs in this domain is a time-consuming and costly process, even with the help of mapping languages such as RML [1, 14, 64]. Indeed, it is still a challenge to automatically generate KGs from multiple semi-structured and textual sources (e.g., descriptions of specific accommodations, reviews, etc.) in order to describe the many facets of this domain, such as the different kinds of accommodations and amenities. Therefore, many KGs in this space are no longer maintained [7, 26] or cannot be easily extended to other tourist destinations [1]. In addition, the relevant ontologies, such as Accommodation Ontology[2], Schema.org[3], and Hontology [9] are to some degree incompatible with each other (as discussed in section 3.3.1) and do not offer a fine-grained representation of some crucial entities (e.g., amenities).

In this paper, we illustrate a general, reproducible, and easily extendable methodology for KG generation and the resulting framework for semi-automatically creating a *Tourism Knowledge Graph (TKG)*, which integrates information from Booking.com, Airbnb.com, DBpedia, and GeoNames. This advanced characterisation of tourism can be used to enable the quantitative analyses of a tourist destination and support several intelligent services. In order to model this data, we developed the Tourism Analytics Ontology (TAO), which offers a more granular characterisation of tourist locations, lodging facilities, and amenities than previous solutions and can be easily reused by similar initiatives.

We showcase our solution by applying it to touristic locations in Sardinia and London, producing over 10M triples describing almost 36K lodging facilities and 898K reviews. The resulting knowledge graph is available online via a SPARQL end-point[4]. The TAO ontology is also available online[5]. Finally, for the sake of reproducibility, we share the code base for our knowledge graph generation pipeline, for engineering TAO, and the evaluation tests[6].

To summarise, the contributions of this manuscript are the following:

- a general data-driven methodology for the semi-automatically generation of knowledge graph that we applied to the tourism domain;
- an open-source pipeline for generating a tourism knowledge graph from (semi-) structured and unstructured data;
- the new Tourism Analytics Ontology (TAO);
- an open-source program to produce the Tourism Analytics Ontology (TAO) using code and data;
- an instance of the tourism knowledge graph (TKG) with data relative to two Tourist Destinations (Greater London and Sardinia island in Italy);
- an evaluation assessing functional, logical, and structural dimensions of TAO and TKG.

[1]The World Tourism Organization (UNWTO) defines in its glossary a *destination* as "the place visited that is central to the decision to take the trip". See https://www.unwto.org/glossary-tourism-terms.

[2]http://ontologies.sti-innsbruck.at/acco/ns.html

[3]https://schema.org/docs/hotels.html

[4]http://tourism.sparql.linkalab-cloud.com/sparql access with login: paper password: journal_p4p3r2022!!

[5]See http://purl.org/tao/ns

[6]See https://github.com/linkalab/tkg

The remainder of this paper is organised as follows. Section 2 describes related works about different knowledge graphs within the tourism domain and methodologies for their creation. Section 3 explains the methodology adopted to guide the knowledge graph creation, detailing the first three iterative phases related to the use cases refinement and ontology design. Section 4 describes the other three phases of the adopted methodology related to the creation of the proposed knowledge graph. Section 5 presents the evaluation, and finally, Section 6 ends the paper with conclusions and future directions of work.

2. Related Work

In this section, we will review the literature on the two main themes concerning this work: i) methodologies for ontology and knowledge graph creation and ii) knowledge graphs within the tourism domains.

2.1. Ontology and Knowledge Graph creation

Creating, maintaining, and further developing knowledge graphs requires the adoption of a number of ontology engineering methodologies (OEMs). Kotis et al. [36] classified such methodologies into three categories: collaborative, non-collaborative, and custom. A collaborative OEM is clearly and systematically defined and involves knowledge engineers, knowledge workers as well as domain experts in all the phases of ontology creation. A non-collaborative OEM does not focus on the collaboration of stakeholders although it still clearly defines phases, tasks, and workflows in a systematic and formal way. A custom OEM does not necessarily define phases, tasks, and workflows in a formal and systematic way; however, it looks for the involvement of communities of practice and the use of tools for the development of ontologies in an agile, decentralized, and most of the time collaborative manner.

There are plenty of works in literature dealing with the creation of knowledge graphs and their methodologies within different domains and constraints [17, 18, 51, 56]. The challenges to be faced depend on such constraints that need to be satisfied by the developers. For example, when knowledge graphs need to be built starting from a complex database schema, there are difficulties (especially related to its dimension) that must be addressed (i.e., how to efficiently read tables, which columns to consider, how to map linked tables, and so on). In this direction, Sequeda et al. [51] presented a novel and unique pay-as-you-go approach to overcome the difficulties of understanding complex database schemas, providing also a use case from a large company. Tamašauskaitė and Groth [56] presented a systematic review of the process for knowledge graph creation. The review methodology aimed at collecting the various steps describing such a process and these include: identification of the data, construction of the knowledge graph ontology, extraction of knowledge, analysis of the extracted knowledge, creation of the knowledge graph and maintenance. The last step is the one that tends to provide periodical updates and edits to the current knowledge graph. In this review, the authors provide suggestions, best practices, and tools supporting the creation and maintenance of knowledge graphs.

In this paper, we present a data-driven methodology that encompasses the semi-automatic generation of the knowledge graph exploiting several off-the-shelf tools and the engineering of a supporting domain ontology using a collaborative OEM (as defined in [36]). The methodology is applied to generate a Knowledge Graph within the tourism domain.

2.2. Knowledge Graphs within the Tourism Domain

In previous years, various attempts have been made to build knowledge bases in several domains, including tourism, using information extracted from websites and social media.

For instance, the 3cixty platform [58] was built during Expo Milano 2015 to create comprehensive knowledge bases, containing descriptions of events and activities, places and sights, transportation facilities, and social activities collected from numerous, local and global data providers, including hyper-local sources. Using the sample platform, in 2016-2017 new knowledge bases have been created for the cities of London, Madeira, and Singapore, as well as for the entire French Cote d'Azur area. The project now seems no longer maintained and no source code was released to recreate the infrastructure. Although a SPARQL endpoint remains active it only allows the user to export data only in HTML and not as RDF.

The Tourpedia platform which was meant to be the DBpedia of tourism was developed within the OpeNER Project [26]. OpeNER (Open Polarity Enhanced Name Entity Recognition) was a project funded under the 7th Framework Program of the European Commission whose main objective was to implement a pipeline to process natural language. The project is no longer maintained although anyone can run the proposed pipeline to view categories, places information, and create and manage events and tour plans for users. Also, on the main website, it is still possible to run the web demo application, showing the sentiment about places through an interactive map. Some datasets are still available for download although other tools, including the SPARQL endpoint, are no longer working.

DBtravel [7] is a tourism-oriented knowledge graph generated from the collaborative travel site Wikitravel that takes advantage of the recommended guidelines for contributors provided by Wikitravel and extracts the named entities available in Wikitravel Spanish version[7] by using an NLP pipeline. As for the previous two projects, the knowledge graph and the source code used to produce it are no longer maintained nor available online.

Other projects demonstrate that semantic technologies and knowledge graphs can be successfully applied to tourism when information is extracted from curated proprietary data sources. In the case of *La Rioja Turismo* Knowledge Graph, Alonso-Maturana et al. [1] retrieve and integrate information referring to attractions, accommodation, tourism routes, activities, events, restaurants, and wineries from heterogeneous and diverse management systems. This approach is focused on the La Rioja Turismo ecosystem but cannot be easily extended to other tourist destinations.

In the case of the Tyrolean Tourism Knowledge Graph [35], data based on Schema.org annotations are collected from destination management organisations (DMOs) and their IT service providers. In this case, the knowledge graph creation is based on the availability of coherent Schema.org annotations in the source websites, which was possible thanks to the cooperation of Tyrolean DMOs. Once again, this scenario is not always applicable because it requires a central organisation to coordinate the different stakeholders.

Another proposed approach was to collect, enrich, and publish Linked Open Data for the Municipality of Catania, a city in Southern Italy, in the context of the project PRISMA, "PlatfoRms Interoperable cloud for SMArt-Government"[8] [11–14]. In this case, Consoli and his colleagues presented the collected city data, described the process and issues to create a semantic data model for emergency vehicle routing and geo-linked data, and discussed a developed prototype. In particular, they described the employed procedures, ontology design patterns, and tools used for ensuring semantic interoperability during the transformation process. Although the project is flexible and can be generalized, the authors did not maintain the resulting knowledge graph.

Other state-of-the-art solutions include the generation of a knowledge graph of tourism in the Chinese language [62, 64]. The authors constructed such knowledge graphs by extracting knowledge from the existing encyclopedia knowledge graph and unstructured web pages in the Chinese language. Besides the fact that this knowledge graph is focused on the Chinese language, the authors focused on semi-structured knowledge extraction and deep learning algorithms to extract high-level entities and relations from unstructured travel notes. The project is no longer maintained and did not provide a SPARQL endpoint.

It is still a big challenge to automatically generate a knowledge graph about tourism that integrates the most important data sources in this field and can be easily extended to other touristic locations. We also lack a single ontology[9] that would offer a fine-grained description of touristic lodging (e.g., Hotel), accommodations (e.g., family room), amenities (e.g., swimming pool), locations (e.g., amusement park), and destinations (e.g., London). The work presented in this paper proposes to address this gap by introducing the Tourism Analytics Ontology (TAO)[10], which offers a granular characterisation of accommodations, tourist locations and destinations[11], and a general, reproducible, and easily extendable pipeline to integrate relevant data sources and generate a knowledge graph

[7]https://wikitravel.org/es/Portada

[8]http://www.ponsmartcities-prisma.it/

[9]We analyse other ontologies in Section 3.3.1.

[10]We describe TAO in detail in Section 3.3 and Appendix B

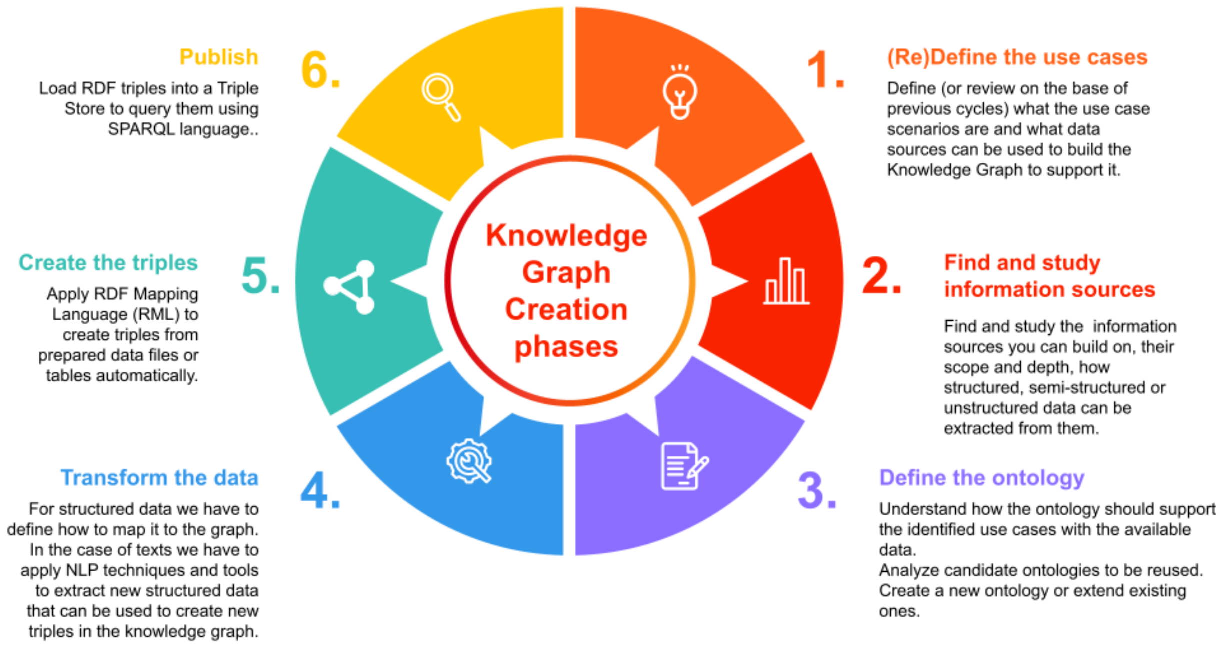
[11]We evaluate in detail how TAO compares to other ontologies regarding the modelisation of relevant tourist entities in Section 5.3.

for the tourism domain. Last but not least, we provide a SPARQL endpoint and plan to periodically update our knowledge graph by using the proposed pipeline. Differently from the approaches discussed above, our proposal can be easily reused and extended to different tourist destinations since we release the full source code, allowing other users to generate new KGs from several data sources that offer worldwide coverage.

3. Methodology for Ontology Design

Our approach for KG construction is aligned with the general methodology analysed in [56] and is organised into six macro phases that can be iteratively repeated to refine the resulting KG. Specifically, the first three phases are the core of a **data-driven design process** that leverages the knowledge embedded in the data sources for guiding the use case refinement and ontology engineering. The last three phases drive the actual implementation of the knowledge graph and its publishing. Figure 1 describes the different phases.

Fig. 1. Tourism Knowledge Graph creation phases


Knowledge Graph Creation phases
1. (Re)Define the use cases
Define (or review on the base of previous cycles) what the use case scenarios are and what data sources can be used to build the Knowledge Graph to support it.
2. Find and study information sources
Find and study the information sources you can build on, their scope and depth, how structured, semi-structured or unstructured data can be extracted from them.
3. Define the ontology
Understand how the ontology should support the identified use cases with the available data.
Analyze candidate ontologies to be reused.
Create a new ontology or extend existing ones.
4. Transform the data
For structured data we have to define how to map it to the graph.
In the case of texts we have to apply NLP techniques and tools to extract new structured data that can be used to create new triples in the knowledge graph.
5. Create the triples
Apply RDF Mapping Language (RML) to create triples from prepared data files or tables automatically.
6. Publish
Load RDF triples into a Triple Store to query them using SPARQL language..

The *first phase* is focused on the definition of the use cases that the knowledge graph should support, that is to say, what are the desired outcomes a user or an application should be able to produce from it. Because our process is driven by what we can find in the data, this is a preliminary definition that is subject to further refinements and that should be revised multiple times until all use cases are positively supported by the KG.

The *second phase* is about understanding how the data at our disposal can support the use cases, but it is also about extracting knowledge from the data to support the ontology definition. On the one hand, the data is used to adapt the use cases to the actual information we have access to, thus extending the scope for some use cases or reducing it for others. For example, if we do not find in the data any information about the total number of rooms for a hotel, we cannot support any use case about the available accommodation capacity for a tourist destination unless we find new data sources. On the other hand, the data is analysed to guide the ontology design. As an example, the accommodations offered on AirBnB have specific types, like shared rooms, which are peculiar to a sharing economy approach. They may also include amenities we seldom find in other forms of hospitality like hotel rooms.

This information incorporates knowledge about the hospitality services for tourism that we can use in the process of ontology design and engineering together with the building of the knowledge graph itself.

The *third phase* focuses on the creation of an ontology to model lodging, tourist destinations, and locations that support all the use cases defined in the first phase and incorporate the domain knowledge distilled in the second phase.

The *fourth phase* is about transforming the data extracted from the data sources in order to prepare it to be used for triple creation in the following phase. During this process, various data wrangling techniques are applied to semi-structured data, whereas natural language processing is applied to unstructured texts (e.g., language detection, named entity extraction, and entity linking).

The *fifth phase* is concerned with triple creation using the data prepared in the previous phase. The triple creation is performed using RDF Mapping Language (RML) in order to include in the knowledge graph also the transformation process metadata.

Finally, the *sixth phase* focuses on the publication of the knowledge graph in a triple store.

The proposed methodology is general and it can be applied whenever it is deemed necessary to design a KG and its supporting ontology with a bottom-up approach. It is well suited to address the need to model a KG to support applications that are based on existing data sources that pose practical constraints to the design and implementation process.

In the following subsections, we describe in detail the first three phases related to use case refinement and ontology creation. In Section 4 we will then describe the final three phases, related to the creation of the knowledge graph.

3.1. Define the use cases

We start with a first general definition of some use cases that we want to cover when building the KG, also considering what data sources could be used to support them. We should also define which kind of applications we would need to implement on top of the KG to support the use cases. This analysis can give us a more general scenario of how the KG would be used. This, in turn, is useful to understand to what extent the data sources can support the scenario and guide the design process on how the KG should be structured. In fact, this phase is intertwined with the second phase (i.e., *Find and study information sources*), discussed in Section 3.2, because we need to consider the information we can extract from the web to support the selected use cases. It is also related to the third phase (i.e., *Define the ontology*) in Section 3.3, because we can have different design approaches regarding the KG depending on what kind of methods and applications it should support (e.g., whether or not we want to apply reasoning techniques on the KG).

In order to generate a KG that can be used to support the analysis of tourist destinations with respect to the supply and demand side, we have identified, together with the domain experts and stakeholders, the following use cases:

(**UC1**) Support the identification of the topics of interest discussed by tourists in their reviews;
(**UC2**) Support the identification of the topics of interest presented in the descriptions of lodging facilities[12] and accommodation[13] offers;
(**UC3**) Support the recognition and linking of tourism entities in the KG for different applications revolving in the domain of social media, news, and blogs;
(**UC4**) Support sentiment analysis [2, 21] applications about tourists toward lodging facilities and destinations;
(**UC5**) Support the classification of tourist destinations on the basis of what they offer and on the basis of tourist opinions.

We also identified a number of applications that can leverage the KG to produce better results (see [41] for a comprehensive overview of applications based on knowledge graphs). In turn, each one of the following applications can be used to better support one or more use cases:

[12]Lodging facilities mean any hotel, motel, motor inn, lodge, and inn or other quarters that provide temporary sleeping facilities open to the public. See https://www.lawinsider.com/dictionary/lodging-facilities)

[13]An accommodation is a place that can accommodate human beings, e.g., a hotel room, a camping pitch, or a meeting room. An accommodation is always part of a lodging facility (e.g., a hotel room is part of a Hotel.)

1. **automatic reasoning**[14] and **graph learning**[15] on the KG allows for the entailment of new triples thus enriching the explicit knowledge other applications can work on; for this reason, it is indirectly related to all use cases;
2. **named entity recognition (NER) and entity linking (EL)** of tourist locations and lodging facilities using the KG have an immediate positive impact on use cases 3 and 5.
3. **relation extraction (RE) in a closed setting** for the tourism industry can be used to support a better understanding of the relations between users and touristic entities thus improving use cases 4 and 5.
4. **tourism-related Topic Modelling** (cluster words/phrases frequently co-occurring together in the tourism context) for texts and documents are written in natural language can be used to support use cases 1 and 2.
5. **tourism-related Topic Labelling** (for clusters of words identified as abstract topics, extract a single term or phrase that best characterises the topic) can also be used to support use cases 1 and 2.
6. **Text Classification** of documents concerning tourism topics can support use cases 1, 2, and 5.
7. **Semantic Annotation** of documents about tourism with entities, classes, and topics based on the KG can be used to support all the use cases by improving user interfaces and user interactions with the textual data.

It is important to note that, the actual feasibility of a use case can be confirmed only when the knowledge graph is built and one or more of the supporting applications are implemented. This validation phase is out of the scope of the present work, which focuses on the design and construction of the knowledge graph.

3.2. Find and study information sources

To support the use cases described in Section 3.1 we need to identify a minimum set of information sources we need throughout the construction of a *core* version of the Tourist Knowledge Graph. After this core Knowledge Graph is created, new information sources could be added by applying the same process described in this work. This is because knowledge graphs have a flexible schema which makes them easily extendable.

Observing the use cases, we can see that we need information sources about:

– lodging facilities and the accommodation they offer;
– user reviews and opinions;
– tourist locations (i.e., points of interest for a tourist such as a train station or a beach);
– tourist destinations such as London or the Costa Smeralda (i.e., the place visited that is central to the decision to take the trip);

The first set of information sources adequately covering the listed items consists of:

– Booking.com, a digital travel company specialised in hotels, B&Bs, and other types of hospitality; from its website we can collect information about accommodations and related offers but also users' opinions expressed as reviews.
– AirBnB, an American company that connects hosts, offering their accommodation spaces (e.g., apartments, rooms, etc.), and travelers, looking for a place to stay; it adopts a peer-to-peer model that originates from sharing economy and represents a new emerging reality in the tourism and accommodation market; its website is a source of information about accommodations and related offers but also users' opinions expressed as reviews.
– DBpedia[16], an open knowledge graph built with structured content extracted from the information created in various Wikimedia projects (e.g., Wikipedia). Specifically, we link entities in TKG to the DBpedia entities of selected classes (e.g., `DBpedia:Places` or `DBpedia:Food`).
– GeoNames[17], a geographical database exposed through APIs and as RDFs documents. We connect entities in TKG with GeoNames entities representing places.

[14]Leveraging Description Logic and OWL.
[15]Using Graph Neural Networks or similar techniques.
[16]https://www.dbpedia.org/
[17]http://www.geonames.org/

It is worth noticing that, although there are many other websites and applications for tourism and hospitality, Booking.com and AirBnB are market leaders and together cover both the traditional accommodation industry and the emerging sharing economy. A similar consideration could be made for DBpedia and GeoNames when we consider places (DBpedia and GeoNames) or general topics related to tourism (DBpedia).

For the present work, we build upon the results of an industrial project about Tourism 4.0 called *Data Lake Turismo* developed by Linkalab s.r.l.[18], which was the evolution of a previous research project promoted by the Digital Innovation Hub of Sardinia[19] and Fondazione Banco di Sardegna[20]. The project aimed at creating a digital platform for tourism data analysis. One of the main components of this platform was a data lake for collecting, transforming, and analysing data in this sector. However, the project lacked a semantic layer that could support and enhance the data analysis, which is the starting point and motivation of the present work.

Through this infrastructure, we have access to data assets related to lodging facilities, user reviews, and opinions; and we enrich them with DBPedia and Geonames.

The data source selection influences both the use case and the ontology definition phases. Although it could be possible to add new data sources to the mix from the beginning, it has a cost and should be postponed wherever possible, because our objective is to complete the construction of a core version of the knowledge graph before expanding its coverage. On the other hand, we should always select data sources that incorporate a rich and well-established model of the business sector (tourism in our case) in the data itself. This is important to support the ontology design with a data-driven analysis process.

3.2.1. Source data exploration

The first step of this phase is to understand what kind of data we can use. We should examine the documentation but we also need to perform an exploratory data analysis on the files and tables accessible in the source data lake in order to have a complete grasp of its contents. This analysis is focused on the following resources available in the data lake:

- data about hospitality:
  - information related to lodging facilities (e.g., hotels, b&bs, resorts) and their characteristics (e.g., name, address, type, hospitality features);
  - information related to accommodations offered by a lodging facility (e.g., hotel room, b&b room, apartment).
  - rent offers for accommodation (e.g., price, number of people, etc.).
- data about user reviews (e.g., user, date, rating, text).

Data is extracted from the data lake in tables with nested structures and needs to be “flattened” to be used by the downstream tasks. This is due to the way the data lake stores information in a redundant and not normalised way.

The result of the exploratory analysis has shown:

- how data is organised in fields and sub-structures;
- that structured and unstructured data (i.e., texts) is available;
- that texts can be in many different languages and it is not always specified in which one;
- that structured data fields can contain numbers, Boolean values, time/date values, or categorical values;
- that data is not always typed and can be represented internally as strings;
- that categorical data is not related to a lookup table or taxonomy;
- that in some cases there are no unique IDs that can be used to identify a resource.

This analysis led us to define some fundamental data pre-processing steps to be executed before building the knowledge graph and the ontology:

[18]Linkalab s.r.l. is an Italian small enterprise specialised in data science and data engineering. Home page https://www.linkalab.it/
[19]https://www.dihsardegna.eu/
[20]https://www.fondazionedisardegna.it/

– **data preparation**: in this step, we extracted the data from the source data lake via SQL queries; next, we stored it on a local file system to be prepared (cleaned, flattened, combined) so that it can be used for downstream tasks.
– **data enrichment**: in this step, we augmented the data using various techniques; specifically, we applied NLP techniques to identify the language of the text (e.g., English, Italian, French, and so on), because downstream tasks depend on it to work properly.

We also found that the data lake source should be integrated with data about attractions and points of interest from other sources. To support this need we identified DBpedia and GeoNames as the most appropriate data sources for the following reasons: i) both sources are stable and constantly maintained, with a vast supporting community; ii) both sources cover the identified destinations (and many others) in depth; iii) both sources are exposed as linked open data and APIs.

3.3. Define the domain ontology

To support the identified use cases and the related applications we want to generate a KG that includes all the relevant entities and their relations. We thus need to define a domain ontology that can model them. To guide the ontology design, we defined a set of functional and non-functional requirements in collaboration with domain experts from Linkalab. We also expressed the same requirements in a more operational form using competency questions, i.e., queries expressed in natural language [27, 47]. In Appendix A, we describe in detail the resulting competency questions as well as both functional and non-functional requirements.

Competency Questions (CQ) are useful as they: i) can be easily understood by non-technical people; ii) can guide the ontology engineering process working as a practical reference of what should be implemented; iii) can be easily tested during the validation process.

We adopted a data-driven design process and followed two complementary approaches when defining the competency questions: i) top-down, by developing new questions with a domain expert and then checking whether they could be answered with our data; and ii) bottom-up, by deriving them from the information available in the source data. Because CQs express all functional requirements in other terms, at the end of this process we could verify that the ontology would successfully model the data in the knowledge graph, which in turn would satisfy the use cases and support the related applications. At the end of this process, we identified the main aspects to model within our domain ontology: i) lodging facilities (buildings), ii) accommodations within lodging facilities, iii) amenities offered to tourists, iv) tourist destinations and locations, and v) user reviews. Next, we analysed several state-of-the-art ontologies covering the tourism domain (detailed in Section 3.3.1), but none of them fully satisfy our requirements. Therefore, we designed and implemented a new ontology, the Tourism Analytics Ontology (TAO), leveraging existing ontologies (e.g., Schema.org, Hontology).

We devote the following subsections to describing: i) the ontologies which we used as a starting point; and ii) the final version of TAO and our design choices.

3.3.1. Reuse of existing ontologies

We analysed several tourism ontologies to assess if they could be reused to support our use cases. We identified three main families of ontologies:

1. ontologies based on Open-Travel or other heavyweight industrial standards, typically focused on information exchange among tourism organisations (e.g., the Harmonise Ontology [24]).
2. ontologies produced by researchers to support specific tasks, such as question answering (e.g., QALL-ME Ontology [49]) and information retrieval (e.g., GETESS [54]) as well as ontologies that combine or build on them (e.g., cDOTT [3], Hontology [9]).
3. ontologies based on Schema.org [28] and GoodRelations [31], such as the Accommodation Ontology.

Based on the functional and non-functional requirements, we then selected three of them: (i) Accommodation Ontology, (ii) the Schema.org markup for hotels, and (iii) Hontology. The latter is currently not available as OWL serialisation at any specific URI and does not seem to be maintained anymore. TAO also reuses other two ontologies:

(iv) GeoNames[21], which is used to specify the geographic locations, and (v) the DBpedia ontology[22], which is used for further characterising locations and food types (e.g., pizza, sushi).

Next, we will describe the selected ontologies and vocabularies and how they have been reused in TAO.

Accommodation Ontology (prefix `acco:`) is an extension of GoodRelations (prefix `gr:`) focused on describing accommodation offers from an e-commerce perspective. It provides additional vocabulary elements for describing hotel rooms, hotels, camping sites, and other forms of accommodations as well as their features. However, it does not make a distinction between the lodging facility (e.g., a hotel as a whole), and the individual accommodations on a lease (e.g., the hotel rooms), because all lodging facility types and accommodation types are sub-classes of the same class (`acco:Accommodation`).

The Accommodation Ontology does not define specific types of amenities (called accommodation features) but "provides a consolidated conceptual model for encoding proprietary feature information". So instead of defining classes for room and hotel features, the ontology provides the generic class `acco:AccommodationFeature` that can hold feature information in varying degrees of formality. A leasing offer is modelled using the GoodRelations relation `gr:Offering` specifying that the offering is a `gr:LeaseOut` using the property `gr:hasBusinessFunction`. Unfortunately, the Accommodation ontology does not cover several concepts that are required for our use cases, including 1) tourist destinations (e.g., London), 2) tourist locations (e.g., beach, church, subway station), 3) tourist reviews.

Schema.org markup for hotels (prefix `schema:`), incorporates and extends many Accommodation Ontology [34] concepts. Schema.org models hospitality according to three main classes[23]:

1. A **lodging business**, (e.g., a hotel, hostel, resort, or a camping site): essentially it represents both the lodging facility, which is the place that houses the actual units of the establishment (e.g., hotel rooms) and the business organisation governing it. The lodging business can encompass multiple buildings but is in most cases a coherent place.
2. An **accommodation**, i.e., the relevant units of the establishment (e.g., hotel rooms, suites, apartments, meeting rooms, camping pitches, etc.). These are the actual objects that are offered for rental.
3. An **offer** to let a hotel room, or other forms of accommodations, for a particular price and a given type of usage (e.g., occupancy), typically further constrained by booking requirements and other terms and conditions.

In this case, we have a clear distinction between lodging business and accommodation because we have two distinct classes: `schema:Accommodation` and `schema:LodgingBusiness`. Unfortunately, Schema.org is not intended to be used as an OWL ontology because its data model is very generic and derived from RDF Schema[24]. The main purpose of Schema.org is to enable sharing of structured data on the Internet whereas OWL is based on formal semantics that enables reasoning on the knowledge graph. In addition, the `schema:LodgingBusiness` class cannot be used in conjunction with GoodRelations ontology without introducing logical contradictions. Specifically, Schema.org defines `schema:LodgingBusiness` as a subclass of `schema:LocalBusiness` which is a subclass of both `schema:Organisation` and `schema:Place`. On the other hand, GoodRelations states that `schema:Organization` and `schema:Place` are disjoint.

We reused Schema.org in TAO by importing and extending a few classes and properties, including `schema:PostalAddress`, `schema:UserReview`,`schema:address`, `schema:subjectOf`. We also selected appropriate Schema.org types that describe places to enrich TAO tourism location classes using `rdfs:seeAlso` to establish a mapping with them[25].

Hontology (prefix `ho:`) is a multilingual ontology for the accommodation sector (H stands for hotel, hostal, and hostel). It is a freely available domain-specific ontology in four languages: English, Portuguese, Spanish, and French [9, 10]. It was partially aligned with QALL-ME and Schema.org and described several useful concepts in this

[21]https://www.geonames.org/ontology/documentation.html

[22]https://www.dbpedia.org/resources/ontology/

[23]Usually called types in Schema.org.

[24]See https://schema.org/docs/datamodel.html

[25]In this respect we can consider TAO ontology an external extension of Schema.org as described in the page https://schema.org/docs/extension.html

domain such as Facilities (a.k.a. amenities), Services, Staff, and Points Of Interest. The ontology is not published as linked data but can be downloaded and used in a local environment. Its latest version dates back to 2012 and therefore it is not aligned with the most recent extensions of Schema.org. In addition, since it is not based on GoodRelations, it does not fulfill our non-functional requirements. We re-implemented within TAO some of its classes describing location amenities, such as `ho:Balance`, `ho:AirConditioning`, `ho:Ballroom`, and `ho:BeautySalon`.

DBpedia Ontology[26] (prefix `dbpedia:`) is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia[27]. The ontology currently covers 685 classes which form a subsumption hierarchy and are described by 2,795 different properties. We used some of the classes from this ontology to enrich TAO tourist location types (subclasses of `tao:TouristLocation`) also mapped to GeoNames geographic features.

GeoNames Ontology[28] (prefix `gn:`) provides elements of description for geographical features, in particular those defined in the geonames.org database. It has three key ontology classes: Feature (a set of all geospatial instances in GeoNames like cities and countries), Class (a set of all feature schemes defined in GeoNames), and Code (a set of abbreviation feature codes in different feature schemes). GeoNames Feature is used for describing concrete geospatial entities (UK, Washington, Colosseum, etc.), whereas GeoNames Class and Code are used for representing meta-information about features. All feature instances are uniquely identified by URI in GeoNames.

We used GeoNames `gn:Feature` class to model classes that are also places (e.g., lodging facilities, tourist locations) and to express their geographic relations using `gn:parentFeature`. We also used GeoNames to enrich TAO tourist location types with specific codes, for example, `tao:Park` was associated to the `gn:L.PRK` code.

3.3.2. The Tourism Analytics Ontology

In this section, we describe the new Tourism Analytics Ontology (TAO) and discuss our design choices. We aimed at developing an ontology i) for which all the requirements listed in Appendix A are fulfilled, ii) that would be able to integrate all relevant information from the data sources, and iii) that would be fully compatible with the Accommodation Ontology, GoodRelations, and Schema.org. Specifically, the Accommodation Ontology is explicitly imported using `owl:imports`, GoodRelations is imported indirectly through Accommodation Ontology and Schema.org is partially included by reusing specific classes and properties or making explicit mappings to it.

The new ontology has the following characteristics:

1. introduces the LodgingFacility class which represents any hotel, motel, inn, or other quarters that provide temporary sleeping facilities open to the public[29];
2. distinguishes between lodging facilities and specific accommodations within lodging facilities;
3. includes an extended hierarchy[30] of lodging facilities types (e.g., hotel, house, resort) ;
4. includes an extended hierarchy of the amenities (e.g., oven, parking garage, baby monitor) offered by lodging facilities;
5. includes an extended hierarchy of geographic features relevant to tourism (based on Schema.org) and enriched with GeoNames feature taxonomy (leveraging the GeoNames mapping[31] data-set);
6. uses Schema.org to model tourist reviews;
7. uses Schema.org to model Tourist Destinations and Tourist Locations;
8. can be easily extended to model other kinds of entities relevant to tourism in the future (e.g., events or restaurants).

[26]https://www.dbpedia.org/resources/ontology/
[27]As defined in the DBpedia ontology page http://web.archive.org/web/20210416134559/http://wikidata.dbpedia.org/services-resources/ontology
[28]https://www.geonames.org/ontology/documentation.html
[29]Definition from Law Insider, see https://www.lawinsider.com/dictionary/lodging-facilities
[30]We use the term "hierarchy" to define a subsumption hierarchy of concepts such as the one used by DBpedia, that is a hierarchy of classes connected with rdfs:subClassOf property.
[31]https://www.geonames.org/ontology/mappings_v3.01.rdf

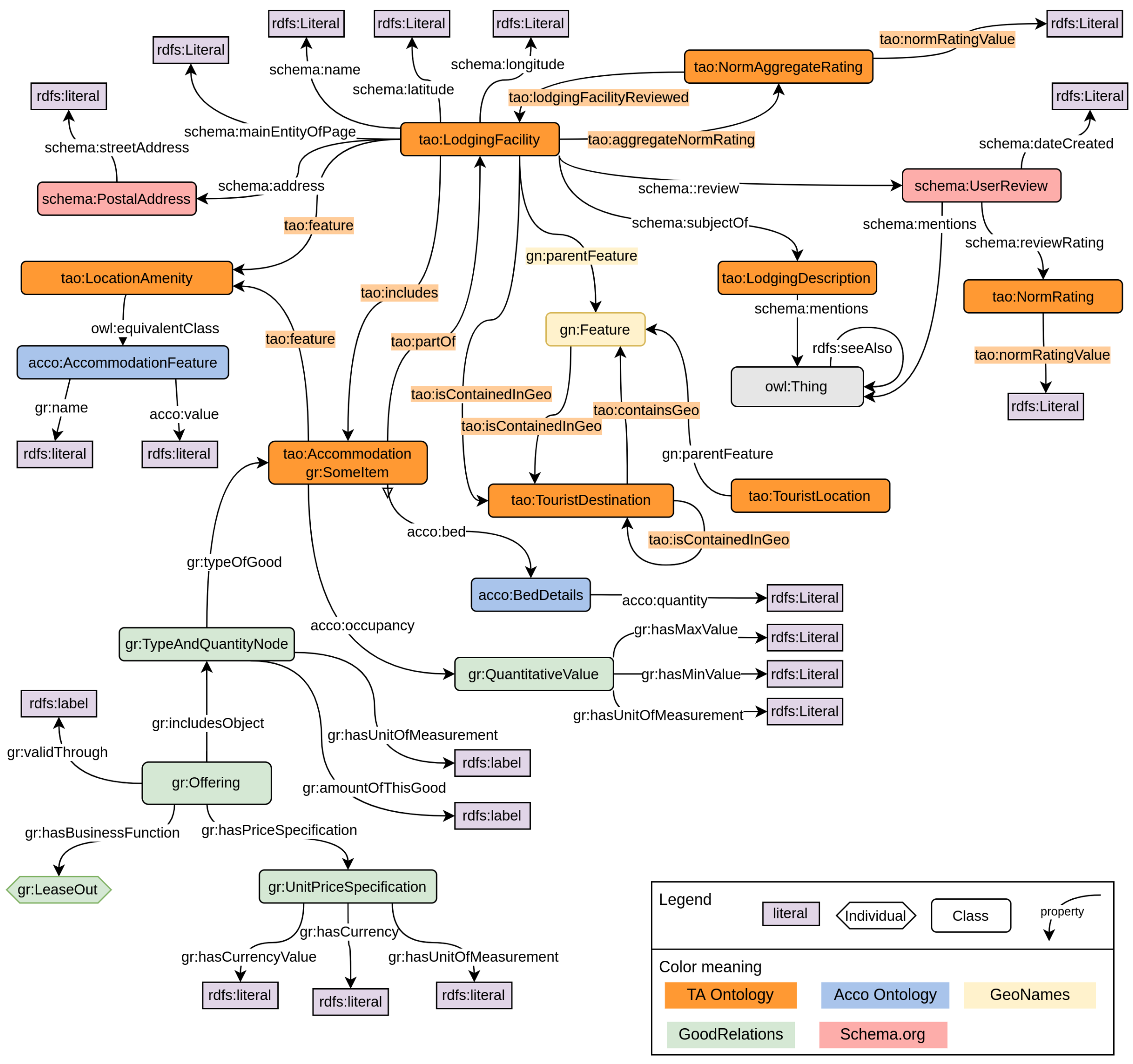


Fig. 2. TAO ontology schema. In this schema, each arrow represents a semantic relationship, starting from its domain and ending in its range.

Figure 2 illustrates the schema of the TAO ontology where the reader can identify the reused classes of the existing ontologies, mentioned above. We will refer to TAO using the `tao:` prefix from now onward. The central classes are `tao:LodgingFacility` and `tao:Accommodation` which are respectively used to model lodging facilities and their accommodations. The `tao:LodgingFacility` class is related to the lodging business concept used in Schema.org, but only refers to the physical place where the accommodations within the facility are located (e.g., a hotel is considered as the building that contains rooms). In this way, there is a clear distinction with the business organisation that governs or owns the lodging facility and no inconsistencies are generated by GoodRelations disjunction between `schema:Place` and `schema:Organization` classes, as discussed in Section 3.3.1. A facility location is described according to its latitude and longitude literal properties and also using the `schema:PostalAddress` class, which favors very detailed specification of the address. To complete the facility description we have literal properties for its name (`schema:name`) and a relevant web page (schema:mainEntityOfPage). We can use the object property tao:aggregateRating[32] to associate a lodging facility to an overall rating, modelled with a node of type tao:NormAggregateRating[33] annotated using the data property tao:normRatingValue to specify a float value between 0 and 1. A lodging facility can also be associated, through the property schema:subjectOf, with a textual description modelled using the tao:LodgingDescription class[34]. Finally, lodging facilities can be connected, using the schema:review property, to one or more user reviews, modelled using the schema:UserReview class. Each review is characterised by the date of creation and associated, using the schema:reviewRating property, with a rating (vote) modelled with a tao:NormRating class[35], that can be used to specify the normalised rating in a specific review. The facility description and the reviews can mention every kind of entity, including those defined in other knowledge graphs (DBpedia and GeoNames) using the schema:mentions property. This information will be typically extracted from the text of descriptions and reviews with various entity linking techniques, such as DBpedia Spotlight [43] (the solution we have chosen for DBpedia), Mordecai [30] (the solution we have chosen for GeoNames), OpenTapioca [16], or Falcon [50]. Entity linking is the task of linking a portion of texts with their corresponding entities in a knowledge graph [45]. These approaches can be used to identify a variety of entities defined in the external knowledge graphs, such as "Eiffel Tower" or "Paris".

The tao:Accommodation class, analogously to schema:Accommodation, represents the actual relevant units of the lodging facility that are offered for rental. It is formally distinct[36] from the physical place where the accommodations are located, which is modelled with the tao:LodgingFacility class instead. TAO uses the tao:includes object property to define the relation between a lodging facility and one of its accommodations. In order for the TAO ontology to maintain a certain degree of compatibility with the Accommodation Ontology, and potentially reuse semantic entities and annotations expressed using it, we defined the tao:Accommodation class as a subclass of acco:Accommodation. In this way, if a node in the KG is a member of tao:Accommodation it is also a member of acco:Accommodation, and all the properties defined in the Accomodation ontology for accommodations are still valid. On the contrary, not all the nodes that are members of acco:Accommodation are also members of tao:Accommodation.

Following GoodRelations best practices, a lease out offering a tao:Accommodation individual is modelled using a combination of GoodRelations classes to define the offering price, type, and quantity:

– the individual is also defined by type gr:SomeItem[37];
– the offering itself is modelled with a node of type gr:Offering, which has an end of validity expressed with the gr:validThrough data property and which is characterised with a specific business function using gr:hasBusinessFunction to specify that is a gr:LeaseOut[38];
– the offering includes the accommodation indirectly through a gr:TypeAndQuantityNode node using the gr:includesObject property and can define its price through a gr:UnitPriceSpecification node;
– a gr:TypeAndQuantityNode node is used to specify which tao:Accommodation node is offered (through the gr:typeOfGood relation), the amount of the good included (using gr:amountOfThisGood data property) and the unit of measure for the amount included (using gr:hasUnitOfMeasurement data property);
– a gr:UnitPriceSpecification node is used to specify the price (using gr:hasCurrencyValue data property), the currency (using gr:hasCurrency data property), and what you are getting for the price (using gr:hasUnitOfMeasurement) i.e., a DAY in the accommodation.

The occupancy accommodation is modelled by using the acco:occupancy property whose value is a gr:QuantitativeValue object, which uses the gr:hasUnitOfMeasurement to specify “C62” lit-

[32]tao:aggregateRating is defined as a subproperty of schema:aggregateRating (relation not shown in Figure 2).
[33]tao:NormAggregateRating is defined as a subclass of schema:AggregateRating (relation not shown in in Figure 2).
[34]tao:LodgingDescription is a subclass of schema:CreativeWork. (relation not shown in Figure 2).
[35]tao:NormRating is defined as a subclass of schema:Rating (relation not shown in Figure 2).
[36]Using owl:disjointWith property
[37]Besides being of type tao:Accommodation
[38]An individual of type gr:BusinessFunction defined in the GoodRelations ontology

eral (used by GoodRelations to indicate "one piece" of something, in this case, a person[39]) as well as the `gr:hasMinValue` and `gr:hasMaxvalue` relations to define the minimum and maximum number of allowed persons. To model an amenity offered by a lodging facility as a whole or as part of a specific accommodation TAO uses the `tao:LocationAmenity` class, which is defined as an equivalent class of `acco:AccommodationFeature` for compatibility with the Accommodation Ontology. It also uses the `tao:feature` property to associate a lodging facility or an accommodation with one or more amenities.

A tourist location (e.g., London's Big Ben or the city of Alghero) is a point or area of interest from a tourist point of view and is modelled with a `tao:TouristLocation` class, which is a subclass of both `schema:Place` and `gn:Feature`. A tourist destination (e.g., Sardinia) is defined as a place that is central to the decision to take the trip and is modelled with a `tao:TouristDestination` class, which is declared as `owl:equivalentClass` of `schema:TouristDestination` and as a subclass of `gn:Feature`. Tourist locations and lodging facilities can be included in a tourist destination using the property `tao:isContainedInGeo`.

For instance, if a tourist destination includes the City of London, all `tao:LodgingFacility` individuals in the City of London (according to `gn:parentFeature` property) are also considered within the same destination. This is because the TAO ontology includes an axiom that defines a chain of properties that state that if `X gn:parentFeature Y` and `Y tao:isContainedInGeo Z`, then `X tao:isContainedInGeo Z`, which can be expressed in functional-style syntax as: `SubObjectPropertyOf( ObjectPropertyChain( gn:parentFeature tao:isContainedInGeo ) tao:isContainedInGeo )`.

TAO includes also several subsumption hierarchies describing the relationships of relevant classes, including:

1. the *lodging hierarchy* with 35 types of lodging facilities (e.g., `tao:Hotel`, `tao:Apartment`, `tao:House`) across 4 levels;
2. the *accommodation hierarchy* with 17 types of accommodations (e.g., `Room`, `EntireApartment`, `Suite`) across 4 levels;
3. the *location amenity hierarchy* with 343 types of amenities (e.g., `Wifi`, `Minigolf`, `Dryer`) across 5 levels;
4. the *tourist location hierarchy* with 146 types of tourist locations (e.g., `City`, `Museum`, `Mountain`) across 5 levels;

Appendix B describes these four hierarchies in more detail.

3.3.3. TAO enrichment

The TAO ontology was produced using a programmatic approach instead of manual editing. Specifically, we developed a building process in Python using the owlready2 [38] library. Compared with other approaches based on templates (OPPL [33], OTTR [53]) or on other languages (like Tawny-OWL [40] which is based on Clojure) we preferred the use of a full programming language like Python which is also very well suited to data manipulation and data transformation. This choice also allowed us to apply well-known software engineering tools and practices and automate some aspects of the ontology building process (e.g., creation of axioms), to version the code instead of just the final ontology, to reduce human errors, and to easily produce inline documentation about the ontology creation process. We also release an open-source version of the Python code that builds the TAO ontology as a Jupyter Notebook[40].

The TAO ontology has to be able to model information derived from typical data sources in the tourism domain, such as Booking.com and AirBnB, which provide (semi)structured data as key/value properties and unstructured data as text regarding lodging facilities, accommodations, amenities, and user reviews. Therefore, we developed a human-in-the-loop strategy, reported in Figure 3, to produce new versions of TAO by continuously enriching the ontology with new types of `tao:LodgingFacility`, `tao:Accommodation` and `tao:LocationAmenity` or new labels for existing types which are derived from the source data. This solution allows us to keep the ontology updated and well-aligned with the actual data.

We start with the basic version of the ontology (orange bullet 1 in the figure), set up external imports, and define classes, properties, and axioms (bullet 2). To further enrich TAO, our ontology engineers in collaboration with

[39]http://www.heppnetz.de/ontologies/goodrelations/v1#UnitPriceSpecification
[40]See https://github.com/linkalab/tkg/tree/main/tao_modelling

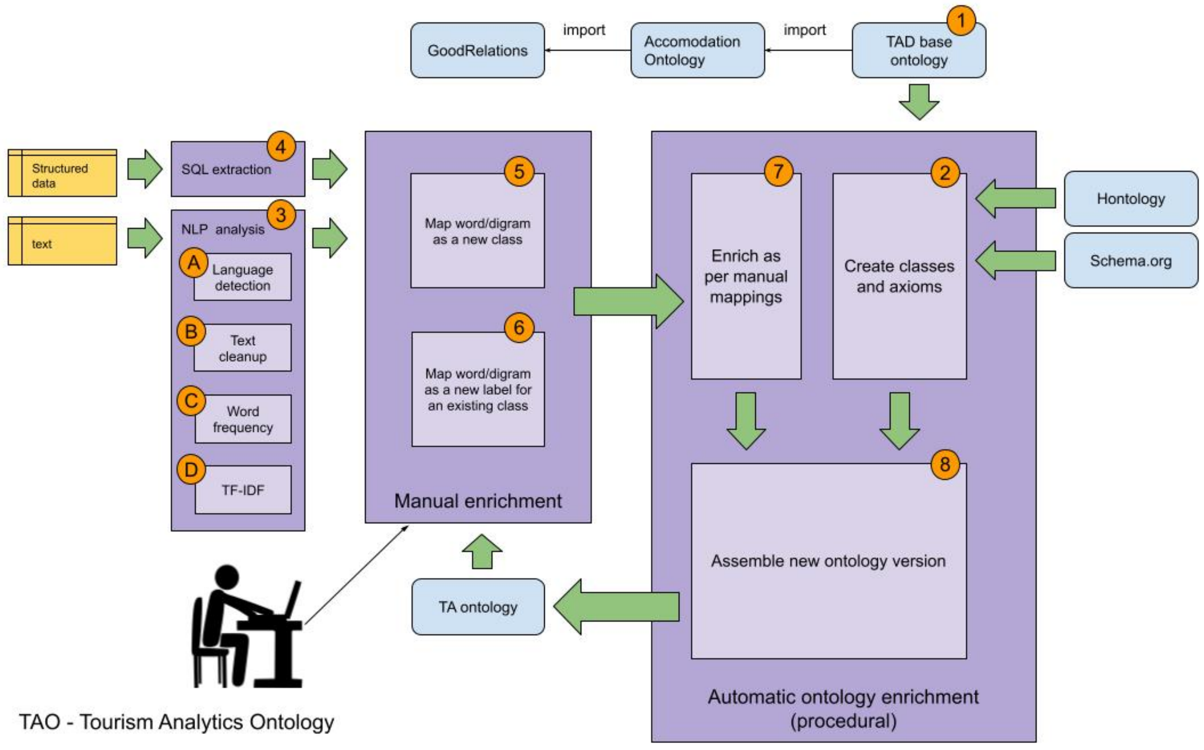

GoodRelations
import
Accomodation Ontology
import
TAD base ontology
1
Structured data
text
SQL extraction
4
NLP analysis
3
A
Language detection
B
Text cleanup
C
Word frequency
D
TF-IDF
5
Map word/digram as a new class
6
Map word/digram as a new label for an existing class
Manual enrichment
7
Enrich as per manual mappings
2
Create classes and axioms
Hontology
Schema.org
8
Assemble new ontology version
Automatic ontology enrichment (procedural)
TA ontology
TAO - Tourism Analytics Ontology

Fig. 3. Ontology enrichment workflow

domain experts, analyse several analytics about the most frequent terms associated with facilities, accommodations, and amenities. Then, they use them to create new relevant classes in the ontology (bullet 5) or add additional labels to an existing class (bullet 6). For example, the mini-golf amenity class was identified in the amenities list extracted from Booking.com, while the holiday home lodging facility alternative label "holiday house" was extracted from AirBnB texts.

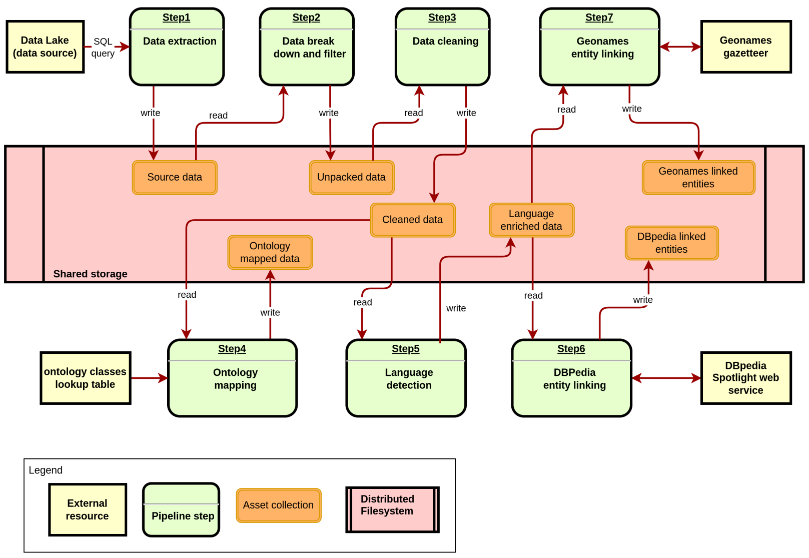
The analytics are produced by two automatic pipelines (3 and 4). The first one processes the unstructured text, extracting and ranking frequent uni-grams and bi-grams from the text descriptions of lodging facilities or user reviews. To achieve this, we relied on Spacy Python library[41] to perform the following sub-tasks: 1) identify language to filter English text only (bullet A), 2) clean the text from special characters (bullet B), 3) perform text frequency analysis (bullet C), and 4) perform TF-IDF analysis (bullet D). The second processes structured data, extracting a list of all possible values for categorical fields that refer to accommodation types, accommodation features, or types of lodging facilities.

Finally, the ontology engineers produce a mapping file that is used (bullet 7) to create new classes, and sub-class relations (using the `rdfs:subClassOf` property) or add labels to existing classes (using the `skos:altLabel` property). We also track the provenance of these changes using the `dc:source` property for classes and the `rdfs:comment` property for labels. The final process (bullet 8) produces a new version of the TAO ontology.

In Appendix E, we report the code snippets for the iterative extension of the TAO ontology.

[41]https://spacy.io/

Fig. 4. High level data transformation workflow diagram



4. Methodology for Knowledge Graph Generation

In the following, we will describe the last three phases for the construction of the Knowledge Graph (depicted in Figure 1).

4.1. Transform the data

The transformation of data is the fourth phase in our approach to building our Tourism Knowledge Graph. Specifically, this phase consists of transforming the information extracted from the data sources into a set of tables, which will be used in the next phase (described in Section 4.2) to produce the actual knowledge graph triples. We devote this section to describing the data transformation process and the technologies for implementing it. Depending on the source data structure and the desired output, we can apply different transformation steps organised as data pipelines. A data pipeline is a series of computational steps organised as a direct acyclic graph where the output of one step becomes the input of one or more downstream steps.

Figure 4 depicts the complete data transformation workflow. Each step can materialise its output (henceforth referred to as *asset*), saving it as a file or storing it in a database application. From the diagram, we can observe four types of components:

1. external resources that are used during the pipeline execution (yellow boxes) representing
   (a) tables in the data lake,
   (b) files mapping text strings to TAO ontology classes,
   (c) DBpedia Spotlight public web service,
   (d) GeoNames gazetteer exposed as an Elasticsearch end-point;
2. pipeline execution steps (green boxes);
3. collections of data assets (files) produced by the execution steps (orange boxes);
4. a distributed file system that stores all the data assets produced and consumed by one or more processing steps (pink box).

At a high level, the workflow consists of 7 steps. The first 3 steps are executed on both structured (key/values) and unstructured (text) data:

1. **Data extraction**: acquires the source data and produces the *Source data assets collection*;
2. **Data break down and filter**: rearranges the data structure and filters out unnecessary data; works in combination with the Data cleaning step and materialises the *Unpacked data assets collection*;
3. **Data cleaning**: reads from the *Unpacked data assets collection*; corrects or removes corrupt, duplicated or inaccurate data; produces the *Cleaned data assets collection*;

The cleaned data is processed differently depending on if it is structured or unstructured. For structured data, the final step is:

4. **Ontology mapping**: uses heuristic rules to identify what ontology class should be used to model each entity described in the data; it produces the *Ontology mapped data assets collection*;

For unstructured data, our objective is to enrich TKG with links from lodging descriptions and user reviews to semantic entities in DBpedia and GeoNames. In this way, TKG would be connected to external knowledge graphs revealing what tourists and business owners are considering important and worth noting. To perform this enrichment we perform entity linking, in three more steps:

5. **Language detection**: identifies the language used in texts to process only English text; produces the *Language enriched data assets collection*;
6. **DBpedia entity linking**: descriptions and reviews texts are processed to recognise and link DBpedia entities; produces the *DBpedia linked entities data assets collection*;
7. **GeoNames entity linking**: descriptions and reviews texts are processed to recognise and link GeoNames entities; produces the *GeoNames linked entities data assets collection.*

In Appendix C, we describe each processing step as well as the employed technological architecture.

4.2. Triples creation

This section presents the fifth phase for the creation of the Tourism Knowledge Graph, shown in Fig. 1, which deals with the creation of the RDF triples. For this, we leveraged the RDF Mapping Language (RML) [19], to build data pipelines for producing RDF triples[42] from text files, and subsequently save them in a serialised format. The RML language is a declarative language used to define how Linked Data is generated from corresponding data sources, using annotations provided through vocabulary terms. RML can use also files as data sources, which is very useful for our scenario. An RML transformation requires the following elements[43]:

1. an RML processor that performs the actual transformation;
2. an input to the RML mapping which is called input data source;
3. an RML mapping, that defines the rules of conversion from any input (structured) data to RDF.

These rules define how to convert an input record (row, XML element, or JSON object) to one or more RDF triples. They are independent of the process of executing the conversion, thus decoupling the implementation from the rules themselves.

In our implementation, we used RMLMapper [20] which is an open-source RML processor developed in Java[44]. We designed different mappings to handle the different sources, i.e., Booking.com and AirBnB.

The output of the RML processor is a set of files containing the RDF triples serialisation in n-quads[45].

To improve the development, debugging and maintenance of RML triple maps we adopted YARRRML [32], a human-readable text-based representation for declarative generation rules[46]. In the following paragraphs, we will examine an example of how a Lodging Facility and all the other related entities can be expressed in TKG by a set of triples created through the process described above. We will represent triples in a graphical form to better understand the knowledge graph structure.

[42]https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-rdf-triple
[43]See https://rml.io/specs/rml/
[44]https://github.com/RMLio/rmlmapper-java
[45]https://www.w3.org/TR/n-quads/
[46]https://rml.io/yarrrml/

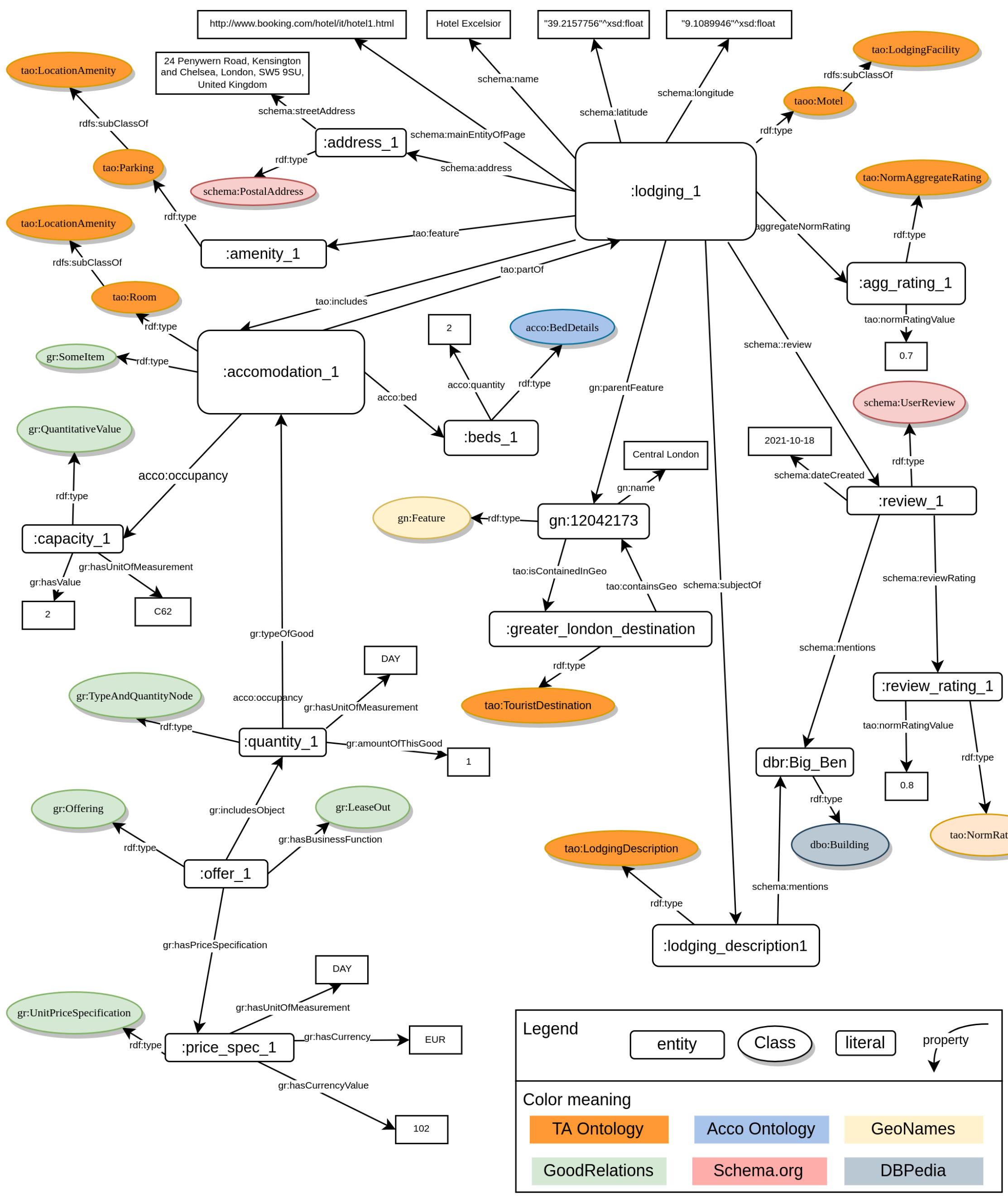


Fig. 5. A high-level example of the main entities used in TKG

4.2.1. *High-level Tourism Knowledge Graph triples structure*

The triples creation process for describing accommodation offers follows the Accommodation Ontology prescriptions and is compliant with GoodRelations and Schema.org best practices. Figure 5 shows an example of TKG structure at a high level. We can observe a lodging facility (`:lodging_1`) that is the subject of a descriptive text (`:lodging_description_1`), that has one review (`:review_1`), and contains one accommodation (`:accommodation_1`). A description is a special kind of creative work (modelled using the `tao:LodgingDescription` class) that can mention one or more real entities like places or food. In the example, the description mentions the Big Ben tower (through the `schema:mentions` property). Also, reviews are considered creative works in Schema.org[47] and are thus related to other real-world entities using `schema:mentions` property. There is an offer (`:offer_1`) to lease out an accommodation that is contained in the lodging facility; `:offer_1` is related to the offered accommodation (`:accommodation_1`) utilizing (`:quantity_1`) node whose properties define what is offered using the property `gr:hasUnitOfMeasurement` (e.g., DAY) and in what quantity using the property `gr:amountOfThisGood` (e.g., 2).

In Appendix D we describe in more detail the structure of triples representing lodging facilities, accommodations, offers, and user reviews in the Tourism Knowledge Graph.

4.3. Knowledge Graph publishing

In this section, we present the triple store publishing TKG, discuss how to identify the different resources in the knowledge graph, and finally how we encoded the provenance. For publishing the knowledge graph we relied on Ontotext GraphDB. The knowledge graph itself is a collection of multiple RDF graphs. Each RDF graph has an associated URI which defines its graph name. For both Booking.com and AirBnB we created two kinds of named graphs:

1. hospitality named graph that contains all the triples created using data assets produced at the and of the ontology mapping step[48] processing semi-structured data extracted from a specific source (e.g., AirBnB) for a certain tourist destination (i.e., London or Sardinia);
2. linked entities named graph that contains all the triples created using data assets produced at the and of the DBpedia or Geonames entity linking steps[49] processing texts extracted from a specific source (e.g., AirBnB) for a certain tourist destination (i.e., London or Sardinia).

A named graph has a custom URI with this structure:
`base_url/tourist_destination/source/enrichment`
Specifically:

1. base_url: `http://tourism.kg.linkalab-cloud.com/ng/`[50]
2. tourist_destination: is used to identify a tourist destination by name (e.g., London or Sardinia)
3. source:
   (a) bkg: is used to identify the source Booking.com ;
   (b) air: is used to identify the source AirBnB;
4. enrichment:
   (a) internal: is used for all the RDF assets that are produced with no entity linking during the transformation phase;
   (b) dbpedia_el: on assets that are enriched with Entity Linking against DBpedia;
   (c) geonames_el: on assets that are enriched with Entity Linking against GeoNames.

As an example, the named graph which is a collection of triples about London hospitality, produced from Booking.com (semi-)structured data (with no entity linking) would have the following URI: `http://tourism.kg.linkalab-cloud.com/ng/london/bkg/internal`.

The use of named graphs implemented as described simplifies the distinction of resources related to a specific tourist destination because we can use the named graphs in SPARQL queries and identify subsets of data through Implicit Graphs using Triple Pattern Fragments[51] (TPF) [60, 61]. This distinction is also useful to express provenance metadata at the named graph level as described in Section 4.3.1.

[47] See https://schema.org/UserReview
[48] as described in Section C.
[49] as described in Section C.
[50] ng stands for named graph.
[51] http://linkeddatafragments.org/

Concerning identifying a resource in the knowledge graph, we use URIs that explicitly contain the external source (e.g., Booking, AirBnB), and the type of resource. The resource URI is structured as follows: `base_url/resource_type/source/unique_id`

In particular:

1. base_url: `http://tourism.kg.linkalab-cloud.com/`
2. resource_type:
   (a) lf: is used to identify Lodging Facility entities;
   (b) ac: is used to identify Accommodation entities;
   (c) of: is used to identify Offering entities;
   (d) rv: is used to identify User Reviews entities.
3. source:
   (a) bkg: the resource is derived from Booking.com;
   (b) air: the resource is derived from AirBnB.
4. unique_id: is an identifier produced by the data transformation phase which is unique for the data source.

As an example, the following URI identifies a lodging facility derived from AirBnB: `http://tourism.kg.linkalab-cloud.com/lf/air/30840569`.

The Tourism Analytics ontology is published as an RDF/XML file at the following URI: `http://purl.org/tao/ns`[52]. To access a specific class or property the hash URI approach is adopted[53] (e.g., `http://purl.org/tao/ns#LodgingFacility` is the URI for LodgingFacility class).

4.3.1. Provenance and dataset metadata

In a dedicated named graph, we loaded also the metadata triples describing the other named graphs and their provenance: `http://tourism.kg.linkalab-cloud.com/ng/meta/prov`. A named graph can be referenced using Quad Pattern Fragments[54] with a URI with the following structure: `base_url?graph=graph_name` where we have:

1. base_url: `http://tourism.ldf.linkalab-cloud.com/graph`
2. graph_name: is the URI associated with the named graph as its name

As an example, the named graph containing the triples about London hospitality produced from Booking.com (semi-)structured data (with no entity linking) would be referenced as:

`http://tourism.ldf.linkalab-cloud.com/graph?graph=http://tourism.kg.linkalab-cloud.com/ng/london/bkg/internal`.

To express the provenance information we used the W3C PROV provenance model. This allows us to track the lineage of data assets produced during the data transformation and triple creation phases following a similar approach as that described in [20] and implemented in the RMLMapper tool. With respect to what is proposed in [20], we applied the Implicit Graphs approach to capture metadata at the Named Graph detail level of granularity, thus generating a minimum number of additional RDF triples for provenance. In this case, the metadata generation time is negligible compared to the overall triple generation time similar to what can be experimented with using the RMLMapper metadata generation feature with a similar configuration.

In PROV we have three main classes:

- `prov:Entity` - a physical, digital, conceptual, or other things with some fixed aspects; entities may be real or imaginary;
- `prov:Activity` - something that occurs over a while and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities;
- `prov:Agent` - something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

Fig. 6. A high-level schema of provenance metadata for a named graph in the Tourism Knowledge Graph.

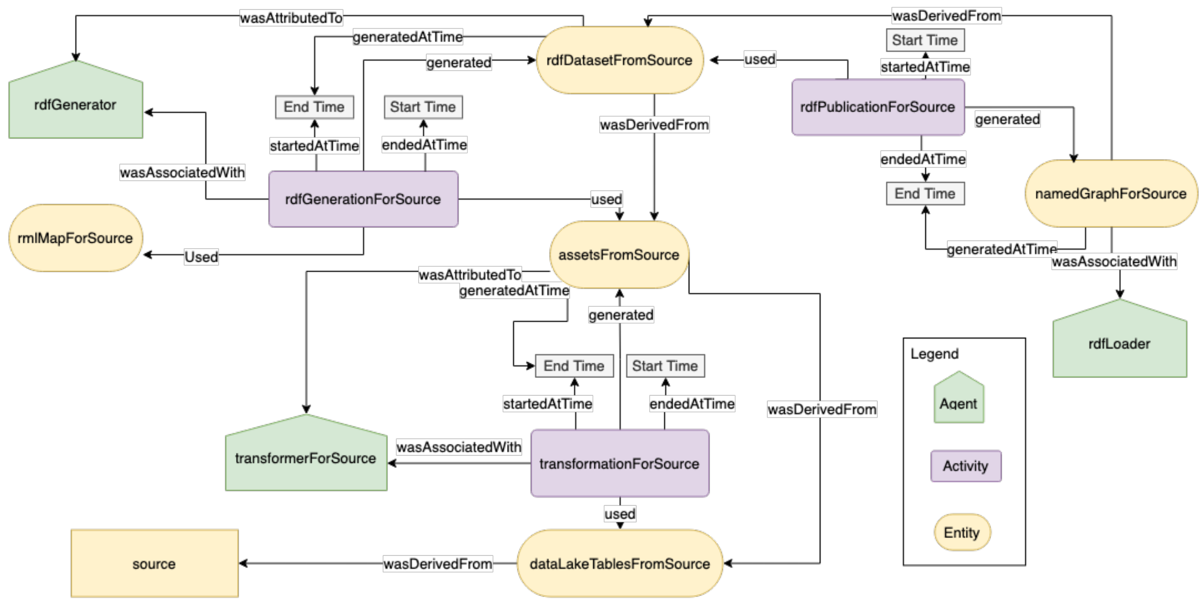


Figure 6 shows a high-level provenance schema describing how provenance metadata for a specific named graph is modelled. Specifically, we can recognise the following PROV entities:

1. `source` - represents the web source for our data (e.g., Booking.com);
2. `dataLakeTablesFromSource` - represents the tables exposed by the data lake containing the data extracted from the source;
3. `assetsFromSource` - represents all the assets created during the transformation phase which are used to produce the RDF triples for a specific named graph;
4. `rmlMapForSource` - represents the RML map document used to produce the RDF triples for a specific named graph;
5. `rdfDatasetFromSource` - represents the RDF graph (serialised as one or more files) that is produced from the source using specific `assetsFromSource` and `rmlMapForSource` entities;
6. `namedGraphForSource` - represents the published named graph.

Moreover, in the same schema, we can identify the PROV Activities involved in the production of a specific named graph:

1. `transformationForSource` - performed to prepare/enrich the data for the triple creation;
2. `rdfGenerationForSource` - performed to produce the triples;
3. `rdfPublicationForSource` - performed to load the triples in the triple store as named graphs.

Finally, we can identify in the schema the following PROV Agents:

1. `transformerForSource` - represents the entire transformation pipeline described in Section 4.1;

[52]This is a redirect to http://schema.linkalab-cloud.com/tao.rdf
[53]See https://www.w3.org/TR/cooluris/#hashuri for an in-depth explanation.
[54]https://linkeddatafragments.org/specification/quad-pattern-fragments/

2. `rdfGenerator` - represents the RML processor software (RMLMapper in our case);
3. `rdfLoader` - represents the agent that loads the RDF graph in the triple store.

The proposed PROV schema can be easily adapted to specify a particular named graph provenance information and can track: (i) when all triples in the named graph are created/updated, (ii) what assets are used to generate the triples, (iii) what RML mapping document was used to generate them. The same can be specified for all the assets produced by the transformation pipeline. The agent entities are also useful to track the software version used to produce each named graph. It is worth noting that, although RMLMapper software is capable of producing provenance metadata, we decided to use the described provenance schema and a custom metadata generator because we wanted to cover all the pipelines (data transformation, RDF generation, and RDF publication) using a common approach and leveraging our orchestration service (Dagster) as described in Appendix C.8.

5. Evaluation

We evaluated TKG and TAO[55] according to functional, logical, and structural dimensions as suggested by previous works [8, 25]. The *functional dimension* refers to the capability of addressing the requirements and offering a useful representation of the tourism domain while the *logical dimension* is about the ability to be successfully processed by a reasoner and produce sound new knowledge. We evaluated both functional and logical dimensions by defining and running a set of tests. We implemented the test cases as RDF files modelled with the TestCase OWL meta-model (prefix `test:`), following Blomqvist et al. [4]. Each test case specifies its inputs, conditions for the execution, the actual testing procedure, and the expected results. All the resulting RDF files are available at https://github.com/linkalab/tkg/tree/main/validation. Finally, the analysis of the *structural dimension* aims at assessing the topological properties of TKG and TAO, which is also compared with other ontologies (i.e., Hontology and Acco). All these analyses provide useful insights on design choices and can be used to iterative refine the knowledge graph. We detail them in the following three subsections.

5.1. Functional dimensions

To verify that the functional requirements are satisfied, we followed the **CQ (Competency Question)[56] verification** approach proposed by Carriero et al. [8]. Specifically, this approach aims at testing whether the competency questions can be answered by running SPARQL queries on the KG. To this purpose, we defined 12 test cases by translating the competency questions, defined in Appendix A.2, into SPARQL queries. The input data were selected from the knowledge graph to test each specific functionality. We used this process to drive the creation and refining of TAO, identifying missing classes or properties and adding them to the ontology. We also used it for verifying that TKG can answer in a meaningful way to all competency questions.

The execution of each test case consists of performing the relative SPARQL query against TKG end point[57]. Queries were manually executed and the results were checked against the expected values. Some CQs required the execution of federated queries to access triples from DBpedia and GeoNames. To this end, we used the `SERVICE` keyword to access Ontotext FactForge SPARQL endpoint[58], which exposes both of them.

All the 12 competency question tests ran successfully. The following example (Listing 5.1) shows a federated SPARQL query that aims to answer "*What are the apartments with wi-fi near at least 2 parks?*"[59].

[55] It is worth noting that TAO has been also verified using OOPS! (https://oops.linkeddata.es/) to find and correct common pitfalls. We manually inspected the results of the tool and, after excluding problems regarding other ontologies or related to incorrect results, we identified and fixed 47 missing annotations, 3 missing domain and range specifications in object properties, 1 wrong equivalent class definition, and 3 inverse relationships not explicitly declared.

[56] A Competency Question is a query expressed in natural language as described in Section 3.3.

[57] To access the SPARQL endpoint use http://tourism.sparql.linkalab-cloud.com/ with username:paper and password:journal_p4p3r2022!!

[58] See http://factforge.net/

[59] In this case a park is considered near the apartment if it is within a distance of 1 km.

```
PREFIX gdb-geo: <http://www.ontotext.com/owlim/geo#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX gn: <http://www.geonames.org/ontology#>
PREFIX tao: <http://purl.org/tao/ns#>
PREFIX acco: <http://purl.org/acco/ns#>
PREFIX schema: <http://schema.org/>
PREFIX onto: <http://www.ontotext.com/>

SELECT ?lodge (SAMPLE(?name) AS ?apartment) (COUNT(?park) AS ?num_parks_nearby)
FROM onto:explicit  ## use only explicit statement without any inference
WHERE {
   { SELECT DISTINCT ?lodge  ?name ?lat ?long WHERE {
            ?lodge a tao:Apartment ; schema:latitude ?lat ;
            schema:longitude ?long ; schema:name ?name; tao:feature ?b.
            ?b  a tao:Wi-FiZone .  } }
    SERVICE <http://factforge.net/repositories/ff-news> {
            ?park gdb-geo:nearby(?lat ?long "1km"); gn:featureCode gn:L.PRK .
    }
}
GROUP BY ?lodge HAVING ( ?num_parks_nearby > 1)
ORDER BY DESC(?num_parks_nearby)
LIMIT 3
```

The query returns the following results.

Lodge	Apartment	Num_parks_nearby
http://tourism.kg.linkalab-cloud.com/lf/bkg/9bd5bef8f50e0e03	"1 Bedroom Luxury Apartment Chancery Lane"	"3"^^xsd:integer
http://tourism.kg.linkalab-cloud.com/lf/air/42701380	"2 bedroom basement apartment with 50 inch TV"	"3"^^xsd:integer
http://tourism.kg.linkalab-cloud.com/lf/bkg/51e2e2d011d57200	"3 Bedroom Palatial Apartment Chancery Lane"	"3"^^xsd:integer

All competency question test cases are available at https://github.com/linkalab/tkg/tree/main/validation/competency_questions[60]

5.2. Logical dimensions

To assess the logical dimension, we first ran a reasoner on TAO and checked for any inconsistency. We then assessed the full TKG according to two strategies suggested in Carriero et al. [8]:

1. **inference verification**, which checks if the inference over the KG produces the expected results (as an example, if a `tao:HotelRoom` accommodation is part of a generic `tao:LodgingFacility` we can infer that the latter is a Hotel);
2. **error provocation**, which aims to provoke an inconsistency error by injecting data that violates the requirements (as an example, an instance of a lodging facility can not be defined of type `tao:Hotel` and `tao:BedAndBreakfast` at the same time).

We thus formulate the relevant test cases to assess what inferences can be performed and what types of errors may be produced by the reasoner. In this case, we can no longer rely on CQs but we have to examine the ontology structure and consider how classes and properties are defined by axioms.

In the following subsection, we will describe more in detail how we conducted these two tests.

[60]We suggest using Protégé for opening the competency questions test cases files.

5.2.1. Inference verification

For evaluating this dimension, we modelled 15 test cases as OWL files using the TestCase OWL meta-model. These files are identified by a unique IRI and contain only the ABox, relying[61] on the TBox of the TAO ontology and the TestCase metamodel[62]. The ABox contains a set of individuals necessary to execute the test and obtain the expected results. All inference verification test cases and the related data sets are available at https://github.com/linkalab/tkg/tree/main/validation/inference_verification.

These tests are useful to understand if the ontology can be successfully used to extend the knowledge graph with reasoning e.g., using inverse properties definitions to materialize backlinks[63], using a chain of object properties to infer new relationships[64], inferring the type of an entity from its properties[65]. For example, let us consider a `LodgingFacility` individual (named `Hotel Splendor`) which is related to `Greater London`, a second-level administrative division defined in GeoNames[66], through the ObjectProperty `gn:parentADM2`. Let us also suppose that there exists a `TouristDestination` individual called `GreatLondonDestination` which includes (via the `tao:containsGeo` property) Greater London. Then, the reasoner should infer that `Hotel Splendor` is also part of `GreatLondonDestination`. It is worth noting that the creation of inference verification tests has been used also during the ontology engineering process for guiding the introduction and refinement of new axioms in TAO.

We performed the final evaluation by loading the test files in Protégé and running the Pellet reasoner[67].

All 15 test cases yielded the expected results.

5.2.2. Error provocation

This test aims at understanding how the knowledge graph (TKG) reacts to the injection of inconsistent data. As an example, since an entity cannot be at the same time a `tao:Hotel` and a `tao:BedAndBreakfast`, we can validate the ontology with regards to this requirement by injecting an individual which is defined as belonging to both classes. The test is successful if the reasoner finds an inconsistency because the appropriate disjointedness axiom is defined in the ontology.

We followed the same strategy used in the inference verification tests described above. In addition, for some tests, we developed also a SHACL file defining further constraints[68].

We implemented 12 test cases for error provocation, testing the identification of wrong patterns in the knowledge graph such as the inclusion of hotel rooms as accommodations in a lodging facility that is not a hotel, the inclusion of accommodation to multiple disjoint lodging facilities, the presence of isolated nodes like a location amenity not connected to any accommodation or lodging facility[69]. We loaded each test file within Protégé, and then we ran both reasoner and the SHACL rules engine[70]. A test is successful if the injected inconsistencies are detected by the reasoner and/or the SHACL validator.

We used this same error provocation technique to test the correct creation of triples during the triple creation process (see section 4.2) and to refine axioms and constraints in TAO.

All error provocation test cases and the related data sets are available at https://github.com/linkalab/tkg/tree/main/validation/error_provocation.

[61]Using `owl:imports`.

[62]http://www.ontologydesignpatterns.org/schemas/testannotationschema.owl

[63]As an example if an Accommodation is `tao:partOf` a lodging facility the inverse relation `tao:includes` can be added to the knowledge graph.

[64]A TouristDestination can be expressed as the composition of other geographic features (using `gn:parentFeature`) so that all lodging facilities contained in those features become also part of the TouristDestination itself.

[65]A lodging facility can be inferred to be of type LowRatedFacility if its normalised rating value is less or equal to a certain value.

[66]See Greater London http://www.geonames.org/2648110/greater-london.html

[67]We used the Pellet reasoner, see the Protégé plug-in https://github.com/stardog-union/pellet/tree/master/protege/plugin

[68]In some tests we use SHACL language to test for integrity constraints that are not limited by the Open World Assumption (OWA)

[69]This case requires the use of SHACL rules because of the open world assumption in OWL.

[70]Using SHACL4Protege Constraint Validator, see https://github.com/fekaputra/shacl-plugin

5.3. Structural dimension

We assessed the structural dimension of TAO and TKG by computing different metrics for assessing ontologies and KG that have been defined and used in the literature [8, 25]. In particular, we followed a similar approach to Carriero et al. [8], which considered both base and topological metrics. Base metrics are used to assess the following quantitative aspects:

- *number of axioms* - the total number of axioms defined for classes, properties, datatype definitions, assertions, and annotations;
- *number of logical axioms* - the number of axioms that affect the logical meaning of an ontology;
- *number of classes* - the total number of classes defined in the ontology;
- *number of object properties* - the total number of object properties defined in the ontology;
- *number of datatype properties* - the total number of datatype properties defined in the ontology;
- *number of annotation assertions* - the total number of annotations in the ontology;
- *DL expressivity* - the description logic expressivity of the ontology.

On the other hand, topological metrics are useful to understand ontology richness, width/depth, inheritance structure, cohesion, and multi-hierarchical degree.

In particular, we adopted the following metrics:

- *Inheritance Richness (IR)* - measures the average number of sub-classes per class[71]. Low values indicate a vertical (deep) ontology whereas high values indicate a horizontal (shallow) ontology.
- *Relationship Richness* - measures the ratio of the number of non-inheritance relationships divided by the number of relationships of all kinds[72]. Values are normalised to one, where 0 indicates that only inheritance relations exist in the ontology and 1 that no inheritance relations are present.
- *Axiom Class Ratio* - measures the ratio of the number of axioms divided by the number of classes[73]. A scarcely axiomatised ontology has a low value of this metric (near zero); higher values are an indication of a better axiomatisation, but very high values can state an excessive axiomatisation.
- *Class/property ratio* - measures the ratio of the number of classes divided by the number of relations[74]. Low values (i.e., ~ 0) are found in ontologies with many properties connecting a few concepts. On the contrary, high values indicate that the ontology has many classes connected by few properties.
- *NoR* - number of root classes (a class which is not a subclass of other classes)[75]. The interpretation of NoR depends on the total number of classes. We expose (i) the ordinal values of NoR and (ii) the ratios between NoR and the number of classes between parenthesis.
- *NoL* - number of leaf classes (all classes that have no sub-classes)[76]. The interpretation of NoL depends on the total number of classes. We expose (i) the ordinal values of NoL and (ii) the ratios between NoL and the number of classes between parenthesis.
- *NoC* - number of external classes[77] defined by [48]. A low value of NoC can indicate that the ontology is semantically independent; a high value can indicate that the ontology depends on concepts defined in other ontologies. The interpretation of NoC depends also on the number of classes in ontology. We expose (i) the ordinal values of NoC and (ii) the ratios between NoC and the number of classes between parenthesis.
- *ADIT-LN* (Average depth of inheritance tree of leaf nodes) - is the average depth of the graph constructed considering classes as nodes and `subClassOf` properties as arcs [78].

[71] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Inheritance_Richness
[72] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Relationship_Richness
[73] https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Axiom_Class_Ratio
[74] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Class_Relation_Ratio
[75] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Number_of_root_classes_.28NoR.29
[76] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Number_of_leaf_classes_.28NoL.29
[77] A class is considered external when it is defined in a different ontology. This metric has been calculated using Protégé.
[78] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Average_depth_of_inheritance_tree_of_leaf_nodes_.28ADIT-LN.29

- *Max breadth* - the maximal value of breadth computed on the graph constructed as for the *ADIT-LN* metric[79]. The value of *Max breadth* should be considered concerning the number of classes in ontology.
- *Average breadth* - the average breadth computed on the graph constructed as for the *ADIT-LN* metric[80].
- *Max depth* - the maximal depth obtained by traversing the graph constructed as for the *ADIT-LN* metric.[81] The value of *Max depth* should be considered concerning the number of classes in ontology.
- *Tangledness* - is the degree of multi-hierarchical classes (which are classes with more than one super-class). It is related to the multi-hierarchical nodes of the graph constructed for the *ADIT-LN* metric[82]. A value of 0 indicates no tangledness; a value of 1 indicates that each class has multiple super-classes.

Tables 1, 2, and 3 report the base and topological metrics measured on TAO, Hontology, and the Accommodation Ontology (Acco). It should be noted that when analysing TAO we considered only the classes and properties defined in this ontology and not the ones imported from other ontologies (i.e., the Accommodation Ontology, GoodRelations). This was done to allow a fair comparison with the Accommodation Ontology, which we import.

All metrics were calculated using OntoMetrics[83] web tool.

Table 1
Base metrics.

Metric name	TAO	Hontology	Acco
# axioms	3960	1453	344
# logical axioms	1237	448	111
# classes	588	284	31
# object properties	19	8	21
# datatype properties	3	31	14
# annotation assertions	2074	682	161
DL expressivity	SROIQ(D)	ALCHQ(D)	ALUH(D)

Table 2
Number of classes by tourism aspect.

Metric name	TAO	Hontology	Acco
Lodging facility types	35	19[1]	5[5]
Accommodation types	17	54[2]	4[6]
Amenities types	343	93[3]	*notprovided*[7]
Tourist location types	146	22[4]	*notprovided*[8]

1. ho:Accommodation sub classes
2. ho:Room sub-classes
3. ho:Facility sub-classes types
4. union of ho:PointOfInterest and ho:Location sub-classes
5. Only selected sub-classes of acco:Accommodation
6. Only selected sub-classes of acco:Accommodation
7. Class acco:AccommodationFeature can hold feature information using acco:value and gr:name data properties to create custom sub-classes.
8. Acco does not model tourist locations.

Table 1 shows that TAO is significantly larger than Hontology and Accommodation Ontology in terms of a number of classes, axioms, logical axioms[84], and annotation assertions. The additional classes mostly describe different types of lodging facilities (35 classes), accommodations (17 classes), amenities (343 classes), and tourist locations (146 classes). Table 2 shows a comparison of the mentioned classes. TAO has more types of lodging facilities, amenities, and tourist locations with respect to Hontology and Acco. Hontology has apparently more accommodation types, but this number may be due to the fact that these types actually combine room types with amenities (e.g., `ho:FamilyRoomWithBalcony`) or the number of beds (e.g., `ho:SingleRoom`, `ho:10BedFemaleDorm`). In this case, we preferred to avoid the addition of specific sub-classes but instead, we used amenities (e.g., `acco:Terrace`) and bed details specifications (using `acco:BedDetails`) to better characterize accommodations.

In terms of properties, TAO introduces only a few new ones, since it reuses most of them from Acco (4), GoodRelations (15), Schema.org (11), and GeoNames (1) as discussed in Section 3.3.2.

[79] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Maximal_breadth
[80] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Average_breadth
[81] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Maximal_depth
[82] See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Tangledness
[83] See https://ontometrics.informatik.uni-rostock.de/ontologymetrics/index.jsp
[84] Logical axioms affect the logical meaning of an ontology. See https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Base_Metrics#Logical_Axiom. On the other hand, non-logical axioms, like entity declarations or annotations, do not affect the consequences of an OWL 2 ontology. See https://www.w3.org/TR/owl2-syntax/#Entity_Declarations_and_Typing

Table 3
Topological metrics.

Metric name	TAO	Hontology	Acco
Inheritance Richness	1.177	0.961	0.742
Relationship Richness	0.413	0.321	0.477
Axiom Class Ratio	6.735	5.116	11.097
Class/propery ratio	0.499	0.706	0.705
NoR	14 (0.02)	17 (0.06)	13 (0.42)
NoL	494 (0.84)	247 (0.87)	23 (0.74)
NoC	19 (0.03)	0 (0.00)	2 (0.06)
ADIT-LN	3.612	2.725	2.439
Max depth	6	5	3
Average breadth	6.578	7.375	5.077
Max breadth	54	29	13
Tangledness	0.179	0.018	0.097

Finally, in terms of expressivity, TAO is similar to Hontology because they share ALCQU features and Acco because they share ALCU features; TAO does not have the H feature because it does not express role hierarchies (SubPropertyOf) as Hontology and Acco; however, TAO has the IS features, indicating the presence of inverse and transitive roles (relations), that the other two ontologies do not have.

The indicators in Tables 1, 2 and 3 can be used to assess and compare TAO, Hontology, and the Accommodation Ontology according to their *transparency*, *flexibility*, and *cognitive ergonomics* [25]. *Transparency* has been defined as "the property of an ontology to be analysed in detail, with a rich formalisation of conceptual choices and motivation". *Flexibility* is related to how easy is to change and evolve the ontology with limited side effects. Finally, *cognitive ergonomics* is the ability of an ontology to be "easily understood, manipulated, and exploited by final users". In the following, we discuss the main indicators of these properties.

TAO performs well according several indicators of transparency [25] as it offers:

- a relative *high number of axioms per class* (6.578). This is higher than Hontology, but lower than Accommodation Ontology, mostly due to the much lower number of classes in the latter;
- a *small coupling* with external ontologies (0.03), similar to Hontology (0) and the Accommodation Ontology (0.06). This is computed as the number of external classes defined in other ontologies (NoC) normalized by the total number of classes. Low coupling allows users to inspect and understand an ontology.
- a *strong cohesion* (i.e., relatedness among classes) due to the low depth of the class hierarchy (ADIT-LN = 3.612), the small number of root classes (NoR = 14), and the high number of leaf classes (NoL = 494);
- a *high inheritance richness* (1.177), which accounts for a more vertical structure, reflecting a more comprehensive coverage of the tourism domain. This is higher than both Hontology (0.961) and Accommodation Ontology (0.742).

The combination of *low coupling* and *strong cohesion* are also indicators of *flexibility* [25].

Finally, TAO exhibits several indicators that are typically associated with a good *cognitive ergonomics*, such as:

- a relatively *low class/property ratio* (0.499), also smaller than Hontology (0.706) and Accommodation Ontology (0.705);
- a *sub-class tree with low depth and breadth* as indicated by ADIT-LN (3.612), max depth (6), and average breadth (6.578);
- a relatively *low tangledness* (0.179 in a range from 0 to 1) that suggests that the inheritance tree has low complexity.

Table 4 reports some statistics about the current prototype of TKG, which includes over 10M triples describing 35K facilities and almost 898K reviews.

Figure 7 shows the distribution of individuals in terms of classes. The most frequent classes are (i) `tao:NormRating` and `schema:UserReview` which are used for reviews; (ii) `acco:AccommodationFeature`[85] that is used as a generic class for amenities together with a specific class from `tao` (e.g., `tao:Kitchen`, `tao:Television`); (iii) the classes used to model an offer such as `gr:Offering`, `gr:TypeAndQuantityNode`, and `gr:UnitPriceSpecification`; (iv) `tao:Accommodation`, `gr:QuantitativeValue`, `gr:SomeItems`, and `acco:BedDetails` are the classes used to model an accommodation; (v) `tao:LodgingDescription`, `tao:LodgingFacility` (and its subclasses), `schema:PostalAdress`, and `tao:NormAggregateRating` that are used to model the lodging facilities. The other classes in the diagram are sub-classes of `tao:LocationAmenity`, `tao:Accommodation` or `tao:LodgingFacility`, which are used to specify precisely their type.

Table 4
Knowledge graph metrics

Metric	**Value**
Number of triples	10,917,081
Number of distinct relations	146
Number of links to DBPedia entities	210,245
Number of unique DBpedia enities linked	3,851
Number of links to GeoNames entities	142,043
Number of unique GeoNames enities linked	3,487
Number of AirBnB reviews entities	358,005
Number of Booking.com reviews entities	539,834
Number of AirBnB LodgingFacility entities	29,870
Number of Booking.com LodgingFacility entities	6,126

6. Conclusions

In this paper, we presented a framework for the semi-automatic construction of a Tourism Knowledge Graph (TKG) and introduced a new ontology for modelling this domain: the Tourism Analytics Ontology (TAO). We have evaluated TKG and TAO according to functional, logical, and structural dimensions.

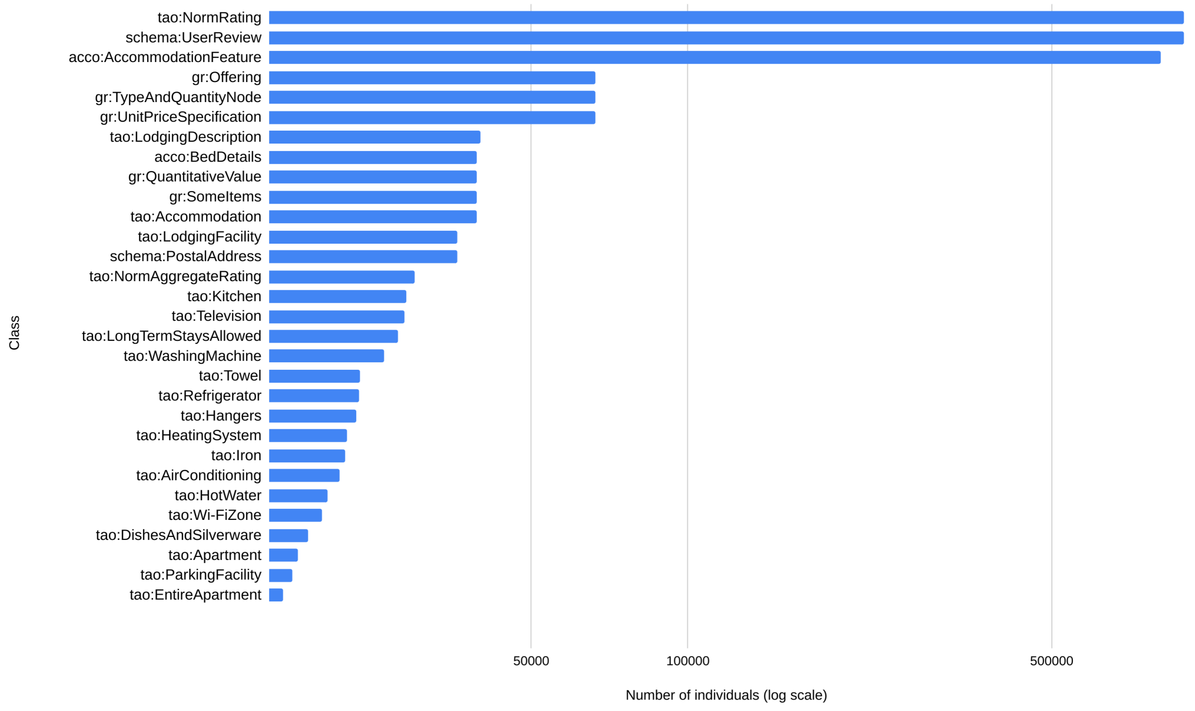
The evaluation suggests that TAO is i) larger than the alternatives (Hontology and Accommodation Ontology) in terms of the number of classes and axioms and ii) also offers higher transparency, flexibility, and cognitive ergonomics.

In future work, we aim to pursue three main pathways. First, we are working on developing NLP solutions to improve the extraction of entities from text, such as descriptions and reviews, so to further enrich the representation of lodging facilities. This step includes the extraction of data from other sources related to several other touristic destinations. Solutions such as Entity Fishing[86] or Open Information Extraction[87] can be leveraged for named entity extraction, including entity detection, name resolution, and named entity recognition.

Second, we want to develop a more scalable solution for integrating data about millions of facilities and users. To achieve such a goal, we will rely on big data frameworks such as Apache Spark and Elasticsearch running in a cluster of machines on cloud computing facilities and we will implement a dedicated DBpedia Spotlight web service to speedup the entity linking process. Third, we want to develop a range of intelligent services based on TKG, including an entity-linking application for automatically annotating accommodations according to reviews and a conversational agent able to answer questions regarding the tourism sector. Knowledge Graph completion will

[85]`tao:LocationAmenity` is defined as an equivalent class to `acco:AccommodationFeature`.
[86]https://github.com/kermitt2/entity-fishing
[87]https://openie.allenai.org/

Fig. 7. Top 30 classes by the number of individuals in the knowledge graph



provide a means to predict relations between entities of the knowledge graph and will be performed by leveraging Knowledge Graph Embedding models (e.g., TransE [5], RotatE [46], ComplexE [59]) or methods based on Graph Neural Networks [65], path-based features [6] and Few-Shot Learning [63].

Transversally to them, we want to extend TAO ontology in order to model other aspects related to tourism, starting with events and restaurants. We also plan to explore other APIs relevant to tourism such as Google Hotel API[88], Google Places API [89] or TripAdvisor[90] to reuse and extend TAO to model also their data in a unified way. To conclude, we are working on automatising as much as possible the pipeline we have used intending to create knowledge graphs with related ontologies in any domain and sources.

Appendix A. Requirements and competency questions

In this section we detail the functional and non-functional requirements identified during the definition of the domain ontology (TAO), described in Section 3.3, as well as the relevant use cases. We also describe the Competency Questions and how they express functional requirements in a more operative form. Finally, we examine the information available in the data sources that supported the formulation of the CQs.

A.1. Requirements

To successfully be used to model a knowledge graph that can support the use cases identified in 3.1, we envisaged that the ontology would need to fulfill the following functional requirements (FR):

[88] See https://developers.google.com/hotels
[89] See https://developers.google.com/maps/documentation/places/web-service
[90] See https://www.tripadvisor.com/developers

Table 5
Mapping knowledge graph's use cases with ontology's functional requirements

Use Case	FR1	FR2	FR3	FR4	FR5	FR6	FR7
UC1 - KG should support the identification of the topics of interest discussed by tourists in their reviews	X	X	X	X		X	
UC2 - KG should support the identification of the topics of interest presented in the descriptions of lodging facilities and accommodation offers	X	X	X	X			
UC3 - KG should support the recognition and linking of tourism entities in the KG for different applications revolving in the domain of social media, news, and blogs	X	X	X	X			
UC4 - KG should support sentiment analysis applications about tourists toward lodging businesses and destinations	X	X	X	X	X		X
UC5 - KG should support the classification of tourist destinations on the basis of what they offer and on the basis of tourist opinions	X	X	X	X	X	X	X

FR 1 model lodging facilities and define a hierarchy[91] of their types (e.g., hotels, hostels, apartments),
FR 2 model accommodations and define a hierarchy of their types (e.g., room, entire apartment, suite);
FR 3 model amenities offered to tourists and define a hierarchy of their types (e.g., disable access, parking garage, baby monitor);
FR 4 model tourist locations (e.g., waterfall, beach, museum, park) and define a hierarchy of their types;
FR 5 model the relations among entities (e.g., geographic relations, mentions, composition/inclusion);
FR 6 model tourist reviews;
FR 7 model tourist destinations (e.g., Sardinia, London), which is the place that is central to the trip.

Functional requirements for the ontology are mapped to the knowledge graph's use cases as described in Table 5. As an example, we can see that since *"KG should support the identification of the topics of interest discussed by tourists in their reviews"* the ontology should model user reviews (FR6) and concepts typically related to what tourists speak about as lodging facilities (FR1), accommodations (FR2), amenities (FR3), and tourist locations (FR4).

When considering non-functional requirements (NFR), the ontology should support reasoning and be based on widely adopted technical and market standards. In particular:

NFR 1 should be defined in OWL[92];
NFR 2 should be based on two *de-facto* standards to model business data:

- Schema.org[93], which is a set of vocabularies developed through a collaborative effort for structuring data on the web. It was originally founded by Google, Microsoft, Yahoo, and Yandex.
- GoodRelations, which is a lightweight ontology for exchanging e-commerce information, namely data about products, offers, points of sale, prices, terms, and conditions, on the Web.

NFR 3 should be easy to extend in order to cover other use cases in the tourism domain.

A.2. Competency questions

Based on the functional requirements we defined the following 12 competency questions:

CQ 1 Which are the first 10 hotels with more than 1,000 reviews and the lowest mean value of users' review scores?

[91]For a description of hierarchies and their implementation in the TAO ontology see Section B.
[92]More specifically it should be based on OWL DL dialect which is designed to provide the maximum expressiveness possible while retaining computational completeness, decidability, and the availability of practical reasoning algorithms.
[93]See https://schema.org/

Table 6
Mapping competency questions with functional requirements

	FR1	FR2	FR3	FR4	FR5	FR6	FR7
CQ1	X					X	
CQ2	X	X	X	X	X		
CQ3	X	X			X		X
CQ4			X	X	X	X	X
CQ5	X			X	X		X
CQ6				X	X	X	
CQ7		X	X	X	X		
CQ8	X				X	X	X
CQ9	X				X	X	X
CQ10	X					X	
CQ11				X	X	X	
CQ12		X			X		X

CQ 2 Find three apartments with Wi-Fi, distant at most 2Km from at least two Parks.
CQ 3 Which Tourist Destinations have the highest percentage of high-priced Lodging Facilities (at least one offer for accommodation for two persons with a nightly price two times over the mean price)?
CQ 4 What are the 10 tourist locations cited most by hotel descriptions that also offer a day Spa in a specific tourist destination?
CQ 5 What are the most cited Tourist Locations in all Lodging Facility descriptions within a certain tourist destination?
CQ 6 What are the Tourist Locations cited most in positive user reviews?
CQ 7 What are the 10 cheapest apartments that offer at least two beds and secured parking and are within 10km from an airport?
CQ 8 Which type of Lodging Facility is more reviewed by tourists in a specific Tourist Destination?
CQ 9 What are the top Tourist Destinations with respect to positive sentiment about food (i.e., percentage of Lodging Facilities with positive reviews that cite food)?
CQ 10 In which months do we have the highest number of user reviews for Hotels?
CQ 11 What Tourist Locations can be found in a Tourist Destination?
CQ 12 How many beds are offered on lease in a certain Tourist Destination?

As we can see, CQs can be more generic or specific depending on which aspect of the ontology we want to describe and eventually test, but all CQs are expressed in terms of questions that can be translated into SPARQL queries against the KG. This is why in some CQs we can use concrete examples (i.e., Wi-Fi) instead of more generic entity classes (i.e., “a location amenity”).

A given competency question usually includes information related to different functional requirements and vice-versa, a certain functional requirement is covered by different competency questions. We can see the mapping between CQs and functional requirements in Table 6.

A.3. Information in data sources

The formulation of the competency questions was also supported by the information available in the data sources. Here, we report a list of the most relevant information available in the data sources (discussed in Section 3.2) that drove the CQs formulation:

1. information about lodging facilities:
   (a) name(s)
   (b) position

   (c) geographic relations with administrative divisions
   (d) geographic relations with tourist destinations
   (e) type (e.g., Hotel, Resort, Motel, B&B, Holiday Accommodations)
   (f) type of accommodation offered (e.g., room, apartment, villa, bungalow, etc.)
   (g) amenities (e.g., sauna, parking, swimming pool, breakfast, air conditioning, etc.)
   (h) accommodation prices exposed on the web
   (i) user ratings
   (j) textual descriptions (to perform Named Entity Recognition, Entity Linking and Relation Extraction, etc.)

2. information about tourist locations:
   (a) name (in multiple languages)
   (b) position
   (c) geographic relations with administrative divisions
   (d) geographic relations with tourist destinations
3. information about tourist destinations:
   (a) name (in multiple languages)
   (b) position
   (c) geographic relations with administrative divisions
   (d) geographic relations with tourist locations
4. tourist reviews about lodging businesses and locations
   (a) user votes
   (b) tourist nationality and type of tourist (family, couple, etc.)
   (c) textual review (to perform Named Entity Recognition, Entity Linking and Relation Extraction, etc.)

This list was employed during the process of ontology engineering, as it helps to define the set of entities and properties that should be modelled by the TAO ontology.

Appendix B. Class hierarchies in TAO

TAO includes several hierarchies of classes connected with rdfs:subClassOf property. This approach was chosen above others (e.g., model taxonomies using SKOS[94]) because we wanted to be compatible with the Accommodation Ontology (where accommodations and amenities types are represented as sub-classes) and simplify the use of Schema.org where class hierarchies are also used. In particular, we have four hierarchies describing the relationships of relevant classes, including:

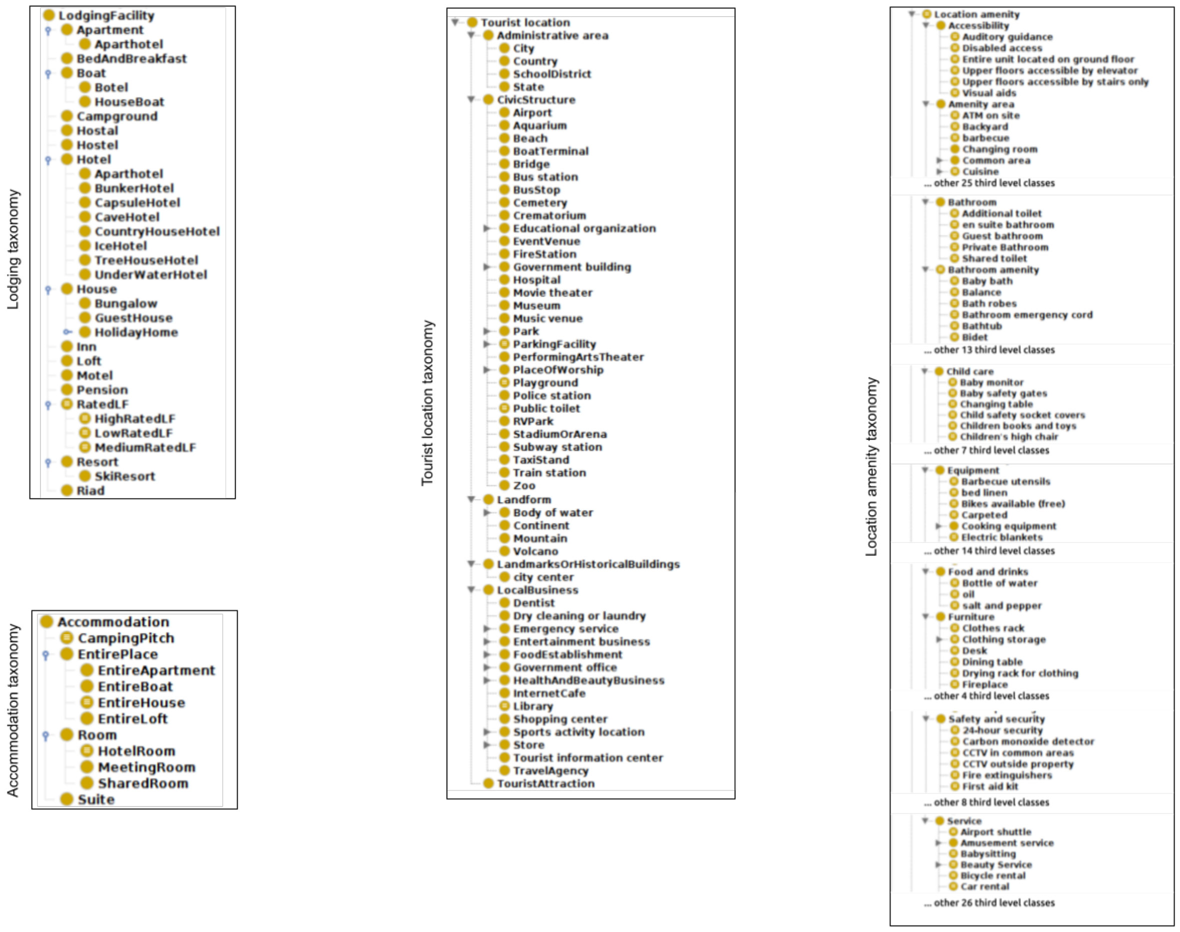
1. the *lodging hierarchy* with 35 types of lodging facilities (e.g., `tao:Hotel`, `tao:Apartment`, `tao:House`) across 4 levels;
2. the *accommodation hierarchy* with 17 types of accommodations (e.g., `Room`, `EntireApartment`, `Suite`) across 4 levels;
3. the *location amenity hierarchy* with 343 types of amenities (e.g., `Wifi`, `Minigolf`, `Dryer`) across 5 levels;
4. the *tourist location hierarchy* with 146 types of tourist locations (e.g., `City`, `Museum`, `Mountain`) across 5 levels;

Figure 8 reports the first three levels of each hierarchy. For each sub-class in a hierarchy we can have one or more of the following implementations:

– if a class is conceptually related to a similar class in other ontologies (e.g., DBpedia), this is modelled with the annotation property `rdfs:seeAlso`;

[94]See https://www.w3.org/TR/2009/REC-skos-reference-20090818/

Fig. 8. A tree representation of the four hierarchies included in the TAO ontology expanded to the third level (some class removed in the location amenity hierarchy for sake of clarity and space).



– if a class is derived from other ontologies, we track the provenance using the `dc:source` property to indicate the original class [95];
– if a class extension[96] is the same as the extension of a class in other ontologies we link them with the `owl:equivalentClass` property [97], or the `rdfs:subClassOf` property if it is narrower[98];
– for each class, we use `rdfs:label` to indicate the primary label and `skos:altLabel` to indicate alternate labels;
– disjoint axioms are added when appropriate to better support the reasoning.

[95]Note that `dc:` stands for Dublin Core ;
[96]The set of individuals that are members of the class.
[97]It is the case of `tao:TouristDestination` which is declared to be `owl:equivalentClass` of `schema:TouristDestination`
[98]It is the case of `tao:EntireApartment` which is declared to be `rdfs:subClassOf` of `acco:Apartment` because in the Accommodation ontology `acco:Apartment` can refer to an apartment as a lodging facility or as an actual accommodation offered on lease.

B.1. Lodging taxonomy

The first hierarchy describes the different types of lodging facilities and their sub-types like in the case of Apartho-tel, which is a special case of a hotel. We also introduce a special case with `tao:RatedLF` and its sub-classes which are used to classify *Lodging facilities* according to their ratings (`tao:NormAggregateRating`). Specifically, `tao:NormAggregateRating` has 3 sub-classes: `tao:LowNormRating`, `tao:MediumNormRating` and `tao:HighNormRating`. These classes can be extended using a data property restriction[99] on `tao:normRatingValue` to implement an automatic classification of a *Lodging facility*.

A rated lodging facility is also part of `tao:RatedLF` (rated lodging facility) class[100] and it can also be inferred whether it is part of one of the following three sub-classes:

– is part of `tao:HighRatedLF` class if it is associated[101] with a `tao:HighNormRating` node;
– is part of `tao:MediumRatedLF` class if it is associated with a `tao:MediumNormRating` node;
– is part of `tao:LowRatedLF` class if it is associated with a `tao:LowNormRating` node;

B.2. Accommodation hierarchy

When modelling *accommodations*, we distinguished two general offerings: (i) entire place (i.e., EntirePlace), and (ii) room (i.e., Room). For these, we also defined sub-classes (e.g., EntireHouse for EntirePlace, HotelRoom for Room). In addition, we modelled two special cases (i.e., CampingPitch and Suite), which are not covered by the general cases. When appropriate, we used equivalence axioms to add useful constraints as in the case of HotelRoom which must be part of one Hotel. Moreover, to support high compatibility between TAO and the Accommodation Ontology, we defined the accommodation classes of TAO as subclasses of the Accommodation Ontology ones (e.g., `tao:CampingPitch` is a subclass of `acco:CampingPitch`).

`tao:CampingPitch` is a subclass of `acco:CampingPitch`) in order to support the best possible compatibility between the two ontologies.

B.3. Location amenity hierarchy

In the case of *location amenities*, we added equivalence axioms to support a certain degree of mapping with how specific accommodation features could be more probably defined using the Accommodation ontology approach[102]. To this end, each sub-class in this hierarchy is also declared as `owl:equivalentClass` to an anonymous class defined in accordance to Accommodation Ontology prescriptions[103]. Thus we define each anonymous class as a subclass of `acco:AccommodationFeature` and as an `owl:intersectionOf` of `owl:Restriction` based on `gr:name` and `acco:value` data properties from GoodRelations. An example is given below in Turtle:

```
tao:AirportShuttle rdf:type owl:Class ;
  owl:equivalentClass [
    rdf:type owl:Class
    owl:intersectionOf (
      acco:AccommodationFeature
      [
        rdf:type owl:Restriction ;
        owl:onProperty acco:value ;
```

[99]See OWL2 specifications https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Data_Property_Restrictions.

[100]Because this class is defined using an existential quantification on the object property `tao:aggregateNormRating` that has some `tao:NormAggregateRating`.

[101]Using `tao:aggregateNormRating` object property

[102]Because there is not a defined taxonomy but a textual label is used to define a specific feature we can only try to guess the label most probably used.

[103]It is defined as “a structured value representing the feature of an accommodation as a property-value pair of varying degrees of formality”; see http://ontologies.sti-innsbruck.at/acco/ns.html#AccommodationFeature

```
        owl:hasValue "yes"@en
       ]
       [
        rdf:type owl:Restriction ;
        owl:onProperty gr:name ;
        owl:hasValue "Airport␣Shuttle"@en
       ]
    ) ;
  ] .
```

In this way, a reasoner can map to the appropriate `tao:LocationAmenity` sub-class an accommodation feature defined using `acco:value` and `gr:name` as prescribed in the Accommodation ontology specifications.

B.4. Tourist location hierarchy

Tourist locations are modelled, whenever possible, according to their respective GeoNames feature codes. This is done by declaring them as `owl:equivalentClass` to an anonymous class which is a restriction on the property `gn:featureCode` that must have an appropriate value from the GeoName feature codes list[104]. An example is given below:

```
tao:Zoo rdf:type owl:Class ;
 owl:equivalentClass [
    rdf:type owl:Restriction ;
    owl:onProperty <http://www.geonames.org/ontology#featureCode> ;
    owl:hasValue <http://www.geonames.org/ontology#S.ZOO>
 ] ;
 rdfs:subClassOf <http://www.geonames.org/ontology#Feature> ;
 rdfs:label "Zoo"@en .
```

Appendix C. Transform the data

In this Appendix, we describe the processing steps for transforming the data shown in Figure 4.

C.1. Data extraction

As the first step, we extracted the relevant data from the source data lake. The extraction process is performed using a SQL big data engine[105]. During this process, the data is also combined and arranged to be more easily processed in the following steps (e.g., unique ids are calculated, and nested columns are exploded). This produces the *Source data* assets collection which consists of:

1. hospitality_supply_assets: containing information about lodging facilities, accommodation, and offers.
2. hospitality_demand_assets: containing information about user reviews.

C.2. Data break down and filter

This second step organizes and structures the information produced in the previous step. Specifically, we need to:

1. break down the information so that we have a distinct asset for each semantic entity we want to model as triples (e.g., lodging facility, accommodation, offer, review);

[104] See https://www.geonames.org/export/codes.html
[105] Amazon Athena, see https://aws.amazon.com/en/athena

2. apply a flat structure to the data, because some columns contain complex data structures as arrays or key/value structures;
3. separate text blobs from the other data preserving their relation to the semantic entity they refer to (e.g., the lodging facility description, the review content).

We can obtain the right structure using specific data pipelines that produce multiple assets out of a single one, flattening the data and filtering out unnecessary columns. This produces an unpacked version of the assets for each source:

1. hospitality_unpacked_supply_assets: containing unpacked information about lodging facilities, accommodation, and offers.
2. hospitality_unpacked_demand_assets: containing unpacked information about user reviews.

C.3. Data cleaning

Here we correct or remove corrupt or inaccurate records from the assets produced in the previous step. In particular, we need to drop duplicated records, remove special characters, normalize categorical fields, normalize date and numeric fields.

From hospitality_unpacked_supply_assets, the Data Cleaning step produces:

1. lodging_assets - containing all structured data relative to lodging facility entities (i.e., entities of type `tao:LodgingFacility`); for each lodging facility a unique ID is produced;
2. lodging_description_assets - containing all descriptions relative to a lodging facility (used to perform Named Entity Extraction and Linking);
3. accommodation_assets - containing all structured data relative to accommodation entities (i.e., entities of type `tao:Accommodation`) in a lodging facility; for each accommodation, a unique ID is produced;
4. offers_assets - containing all structured data relative to accommodation offers (i.e., entities of type `gr:Offering` that will be modelled as prescribed by the Accommodation Ontology); for each offer, a unique ID is produced;
5. amenities_assets - containing all accommodation features (a.k.a. amenities) that are related to a lodging facility and/or to accommodation.

Instead, from hospitality_unpacked_demand_assets, the Data Cleaning produces:

1. reviews_assets - containing all structured data relative to user reviews about a lodging facility; for each review, a unique ID is produced;
2. reviews_content_assets - containing all text content for user reviews about a lodging facility (used to perform Named Entity Extraction and Linking);

C.4. Ontology mappings

At this stage, we identify and map the classes of the structured data to transform them into triples.

For instance, if a lodging business is represented as a record like:

hotel_id	name	structure_type
9f40f613d308cf80	Chelsea BnB	Bed and breakfast

after the ontology mapping step, a new field lf_class (lodging facility class) is added with the “BedAndBreakfast” class name:

hotel_id	name	structure_type	lf_class
9f40f613d308cf80	Chelsea BnB	Bed and breakfast	BedAndBreakfast

Structured data include categorical columns that refer to concepts in the TAO ontology. In particular, there are three hierarchies in the ontology (See Appendix B for details) that we have to reconcile with categorical columns in the data:

1. lodging facility types: for each lodging table record we have a text field that contains the name of the lodging facility type; this field can be used to associate the correct `tao:LodgingFacility` subclass to the individual lodging facility the record is about;
2. accommodation types: for each accommodation table record we have a text field that contains the name of the accommodation facility type; this field can be used to associate the correct `tao:Accommodation` subclass to the individual accommodation the record is about;
3. accommodation features (amenities) types: for each amenity table record we have an accommodation feature associated with a specific lodging facility (via an external key ID that refers to the lodging table). This field can be used to associate the correct `tao:LocationAmenity` subclass to the individual amenity the record is about.

To perform the reconciliation we use a heuristic process based on rules that can identify the most appropriate class to use to model an entity. The heuristic process uses lookup tables extracted from the ontology where we have each class associated with each of its labels. In this way, we leverage the ontology enrichment we already described in Section 3.3.3. The reconciliation is thus performed by adding the correct class name in a new column of the data table so that it can be used during the triple-creation phase. The ontology mapping step produces new types of assets that are part of the *Ontology mapped data* asset collection:

1. classified_lodging_assets;
2. classified_accommodation_assets;
3. classified_amenities_assets.

These assets will be fed into the triple creation process.

C.5. Language detection

This step applies a language detection algorithm [52] to the text contained in the lodging description and reviews content tables. The detected language is used to enrich *lodging_description_assets* and *reviews_content_assets* with a new language column so that subsequent steps can process only English texts. The enriched assets are part of the *Language enriched data* asset collection.

C.6. DBpedia entity linking

To perform the Entity Linking task against DBpedia we have applied DBpedia Spotlight [15, 42] APIs[106] to the English text contained in the lodging description and reviews content tables. DBpedia Spotlight identifies and annotates entities based on the following pipeline process:

– Spotting: identifies possible entity mentions (surface forms) from the original input text.
– Candidate selection: selects the DBpedia resources that are candidate meanings for each surface form.
– Disambiguation: determines which candidate is the most likely resource for each surface form.
– Filtering: adjusts the annotation task based on the user requirements.

For the filtering step, we restricted the annotation scope to the following type of entities: `DBpedia:Activity, DBpedia:Food, DBpedia:Holiday, DBpedia:MeanOfTransportation, DBpedia:Place, Schema:Event, Schema:Place`. The result of the DBpedia entity linking process produces two new types of assets which are part of the *DBpedia linked entities* asset collection:

1. lodging_dbpedia_linked_assets - containing a record for each DBpedia entity linked to a lodging facility identified by its unique ID;

[106]https://www.dbpedia.org/resources/spotlight/

2. review_dbpedia_linked_assets - containing a record for each DBpedia entity linked to a user review identified by its unique ID.

We used these assets in the triple-creation process.

C.7. GeoNames entity linking

This step performs an Entity Linking task against GeoNames so that places named in the lodging descriptions or the reviews are linked to the GeoNames corresponding entities.

To this end, we employed an open-source software called Mordecai[107] [29], a full-text geoparsing system that extracts place names from the text, resolves them to their correct entries in a gazetteer, and returns structured geographic information for the resolved place name. Mordecai is based on a language-agnostic architecture that uses word2vec [44] for inferring the correct country for a set of locations in a piece of text. As a gazetteer, it uses a custom-built Elasticsearch database populated with GeoNames data. Mordecai is integrated within the Spacy library[108]. Analogously to what is described in Section C.6 for DBpedia, we used Mordecai to process all English text contained in the lodging description and review content tables. The result of the GeoNames entity linking process produces two new types of assets which are part of the *GeoNames linked entities* asset collection:

1. lodging_geonames_linked_assets - containing a record for each GeoNames entity linked to a lodging facility identified by its unique ID;
2. review_geonames_linked_assets - containing a record for each GeoNames entity linked to a user review identified by its unique ID.

We used these assets in the triple-creation process.

C.8. Implementation strategy

To support the data transformation described in the previous sections, we identified the following requirements for our technological architecture:

- Data-driven,
- Flexible and easily extensible,
- Scalable in a distributed computing environment,
- Easily manageable,
- Easily instrumented for lineage (a.k.a. provenance) metadata collection.

Following the requirements, the data computation is organised using the pipeline approach already described. This approach is optimal to create a distributed computation if the intermediate and final materializations are stored on a distributed file system. This is the same approach adopted by Apache Spark and other big data frameworks.

To manage the execution of a set of data pipelines, we used Dagster[109], an open-source orchestrator service. Dagster can be deployed on a single machine or a distributed environment like Kubernetes or AWS Elastic Container Service clusters. Thanks to this flexibility we started using a single machine to simplify the deployment process, without losing the opportunity to switch to a distributed architecture in the future. Dagster can also expose metadata about the execution of each pipeline and the produced assets, enabling our system to generate provenance information for the Knowledge Graph. The data transformation code is developed using Python Pandas[110] library. We released the pipelines built on Dagster as an open-source resource for the paper[111].

[107]https://github.com/openeventdata/mordecai
[108]Only Spacy v2.x is supported at the moment
[109]https://dagster.io/
[110]https://pandas.pydata.org/
[111]See https://github.com/linkalab/tkg/tree/main/kg_pipelines

C.9. Performance on a single server

We used a single node with CPU AMD Ryzen™ 7 5800H, 32GB of RAM, 1TB SSD, and Ubuntu 20.04. With this setup the data transformation over the booking.com and Airbnb data was about 8 hours and 45 minutes, where the entity linking process took 7h 14m, language detection 1h and 26m, leaving all the other data extraction and transformation steps only 14 minutes of execution time. This is because entity linking is performed invoking DBpedia Spotlight public end-points so that we could only apply a limited concurrency on the requests to the external web service to avoid server-side errors. This is the main limitation to scalability for the present implementation because the other data processing steps are very fast being executed using a big data query engine for the extraction (Amazon Athena) or using Python pandas with all data loaded in RAM. If a higher entity linking speed is needed it is possible to create a self-managed setup for DBpedia Spotlight as described in their website[112]. Regarding language detection, it can be optimised in a single-node setup using a multithreading approach similar to what has been implemented for entity linking and can also scale horizontally on multiple nodes because it only requires local CPU time.

We reduced the used disk space using Parquet files for tabular data. The total storage space was 3GB which can be reduced to 1.6GB if all triple files are compressed. To support storage scalability a distributed filesystem could be used as suggested in Appendix C.8.

Appendix D. Triple structure details for TKG

In this Appendix, we describe the structure of triples representing lodging facilities, accommodations, offers, and user reviews in the Tourism Knowledge Graph. We refer to Figure 5 in the following sections.

D.1. Lodging facility entities triple structure

In Figure 5, we can steer our focus to observe triples modelling a lodging facility, which includes:

1. an address entity (`:address_1`), modelled as a `schema:PostalAddress` class that gives us great flexibility to define the facility position;
2. one or more accommodation features entities that are associated with the lodging facility using the `tao:feature` property; in our example, we have the node `:amenity_1` of type `tao:Parking`[113].
3. an aggregated rating entity (`:agg_rating_1` in our example) that is used to model the overall user rating for the lodging facility (which is related to the ratings expressed by the single users' reviews) that specifies the vote in a normalised range from 0 to 1.

D.2. Accommodation entities triple structure

Accommodation is always related to a lodging facility, in compliance with the Accommodation ontology, and it includes:

1. its maximum and minimum occupancy capacity, using a `gr:QuantitativeValue` node (`:capacity_1` in our example);
2. its provision of beds, using an `acco:BedDatails` node (`:beds_1` in our example);
3. the type of accommodation[114] (using one of the TAO ontology classes like `tao:Room`).

[112]See http://dev.dbpedia.org/Dbpedia_Spotlight

[113]In general the class of the amenity should be the most appropriate TAO ontology class among all the subclasses of `tao:LocationAmenity` as detected during the Ontology mapping step described in Section C.4

[114]As detected during the Ontology mapping step described in C.4

D.3. Offer entities triple structure

We describe a commercial offer for leasing out an accommodation leveraging GoodRelations. As shown in Figure 5 an offer can be expressed in terms of:

1. a node (`:quantity_1`) of type `gr:TypeAndQuantityNode` used to specify the number of days it is offered using `gr:amountOfThisGood` and `gr:hasUnitOfMeasurement` properties;
2. a node (`:price_spec_1`) of type `gr:UnitPriceSpecification` used to specify the price and currency for each day using the `gr:hasUnitOfMeasurement`, `gr:hasCurrency` and `gr:hasCurrencyValue` properties.

D.4. User reviews triple structure

A user review of the lodging facility is represented in TKG by two entities:

1. a node (`:review_1`) of type `schema:UserReview` with a `schema:dateCreated` property used to specify the review creation date;
2. a node (`:review_rating_1`) of type `tao:NormRating` that is used to specify the actual rating normalised to 1 (using `tao:normRatingValue`) property.

Appendix E. TAO Extension

Here we report an example of the Python code we implemented on top of owlready2 for extending the TAO ontology with new classes:

Listing 1: Python snippet to extend the TAO ontology with new classes.

```python
from owlready2 import *
world = World()
tao_ontology = world.get_ontology("./ontologies/tao_base.rdf").load()
tao = tao_ontology.get_namespace("http://purl.org/tao/ns#")
with tao:
    class TouristLocation(schema.Place, gn.Feature):
        label = [locstr("Tourist␣location", lang = "en")]
        comment = """A location is a point or area of interest from a tourist point of view,
            which a particular product or service is available, e.g. a museum, a beach, a bus
            stop, a gas station, or a ticket booth. The difference to gr:BusinessEntity is that
            the gr:BusinessEntity is the legal entity (e.g. a person or corporation) making
            the offer, while tao:Location is the store, office, or place. A chain restaurant
            will e.g. have one legal entity but multiple restaurant locations. Locations are
            characterized by an address or geographical position and a set of opening hour
            specifications for various days of the week."""
        altLabel = [locstr("Point␣of␣interest", lang = "en"), locstr("Area␣of␣interest", lang =
             "en"), locstr("Location", lang = "en")]
        seeAlso = gr.Location
    class TouristDestination(gn.Feature):
        label = [locstr("Tourist␣destination", lang = "en")]
        comment = """A tourist destination. A TouristDestination is defined as a Place that
            contains, or is colocated with, one or more TouristLocation and LodgingFacility,
            often linked by a similar theme or interest to a particular tourist audience. The [
            UNWTO](http://www2.unwto.org/) defines Destination (main destination of a tourism
            trip) as the place visited that is central to the decision to take the trip."""
        equivalent_to = [schema.TouristDestination]
tao_ontology.save(file = "output_ontology/tao_new.rdf", format = "rdfxml")
```

In the following code snippet we show an example of how we can process a CSV file that describes new classes to be integrated into the ontology. All data from the CSV file are loaded in a pandas dataframe and processed by a custom function (process_entity function) that uses owlready2 to handle OWL class creation or modification. For more detail see the full source code.

Listing 2: Example of massive creation of classes and labels using Pandas library in Python.

```
from owlready2 import *
import pandas as pd
world = World()
tao_ontology = world.get_ontology("./ontologies/tao_base.rdf").load()
tao = tao_ontology.get_namespace("http://purl.org/tao/ns#")
df = pd.read_csv("./enrichment/booking_facilities.csv")
df.apply(lambda r: process_entity(
    [tao_solo, acco], r['entity'],r['parent_class'],r['class'], r['type'], r['is_amenity'],
    provenance = "Booking.com␣features␣lists␣extraction.",
    comment_text = "Enriched␣Booking.com␣features␣lists␣extraction"), axis=1)
tao_ontology.save(file = "output_ontology/tao_new.rdf", format = "rdfxml")
```

References

[1] R. Alonso-Maturana, E. Alvarado-Cortes, S. López-Sola, M. O. Martínez-Losa, and P. Hermoso-González. La Rioja turismo: The construction and exploitation of a queryable tourism knowledge graph. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11153 LNCS:213–220, 2018. ISSN 16113349. .

[2] M. Atzeni and D. R. Recupero. Multi-domain sentiment analysis with mimicked and polarized word embeddings for human-robot interaction. *Future Gener. Comput. Syst.*, 110:984–999, 2020. . URL https://doi.org/10.1016/j.future.2019.10.012.

[3] R. Barta, C. Feilmayr, B. Pröll, C. Grün, and H. Werthner. Covering the semantic space of tourism : An approach based on modularized ontologies. *ACM International Conference Proceeding Series*, 2009. .

[4] E. Blomqvist, A. Seil Sepour, and V. Presutti. Ontology testing - Methodology and tool. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7603 LNAI(November 2020):216–226, 2012. ISSN 03029743. .

[5] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

[6] A. Borrego, D. Ayala, I. Hernández, C. R. Rivero, and D. Ruiz. Cafe: Knowledge graph completion using neighborhood-aware features. *Engineering Applications of Artificial Intelligence*, 103:104302, 2021. ISSN 0952-1976. . URL https://www.sciencedirect.com/science/article/pii/S0952197621001500.

[7] P. Calleja, F. Priyatna, N. Mihindukulasooriya, and M. Rico. DBtravel: A tourism-oriented semantic graph. In C. Pautasso, F. Sánchez-Figueroa, K. Systä, and J. M. Murillo Rodríguez, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11153 LNCS of *Lecture Notes in Computer Science*, pages 206–212, Cham, 2018. Springer International Publishing. ISBN 9783030030551. . URL http://link.springer.com/10.1007/978-3-030-03056-8.

[8] V. A. Carriero, A. Gangemi, M. L. Mancinelli, A. G. Nuzzolese, V. Presutti, and C. Veninata. Pattern-based design applied to cultural heritage knowledge graphs. *Semantic Web*, 12(2):313–357, 2021. ISSN 22104968. .

[9] M. S. Chaves and C. Trojahn. Towards a multilingual ontology for ontology-driven content mining in Social Web sites. *CEUR Workshop Proceedings*, 687, 2010. ISSN 16130073.

[10] M. S. Chaves, L. Freitas, and R. Vieira. Hontology: A multilingual ontology for the accommodation sector in the tourism industry. *KEOD 2012 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, pages 149–154, 2012. .

[11] S. Consoli, A. Gangemi, A. G. Nuzzolese, S. Peroni, V. Presutti, D. R. Recupero, and D. Spampinato. Towards emergency vehicle routing using geolinked open data: the case study of the municipality of catania. In A. Gangemi, H. Alani, M. Nissim, E. Cambria, D. R. Recupero, V. Lanfranchi, and T. Kauppinen, editors, *Joint Proceedings of the 1th Workshop on Semantic Sentiment Analysis (SSA2014), and the Workshop on Social Media and Linked Data for Emergency Response (SMILE 2014) co-located with 11th European Semantic Web Conference (ESWC 2014), Crete, Greece, May 25th, 2014*, volume 1329 of *CEUR Workshop Proceedings*, pages 31–42. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1329/preface-SM.pdf.

[12] S. Consoli, A. Gangemi, A. G. Nuzzolese, S. Peroni, V. Presutti, D. R. Recupero, and D. Spampinato. Geolinked open data for the municipality of catania. In R. Akerkar, N. Bassiliades, J. Davies, and V. Ermolayev, editors, *4th International Conference on Web Intelligence, Mining and Semantics (WIMS 14), WIMS '14, Thessaloniki, Greece, June 2-4, 2014*, pages 58:1–58:8. ACM, 2014. . URL https://doi.org/10.1145/2611040.2611092.

[13] S. Consoli, A. Gangemi, A. G. Nuzzolese, S. Peroni, D. R. Recupero, and D. Spampinato. Setting the course of emergency vehicle routing using geolinked open data for the municipality of catania. In V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*, volume 8798 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2014. . URL https://doi.org/10.1007/978-3-319-11955-7_4.

[14] S. Consoli, V. Presutti, D. R. Recupero, A. G. Nuzzolese, S. Peroni, M. Mongiovì, and A. Gangemi. Producing linked data for smart cities: The case of catania. *Big Data Res.*, 7:1–15, 2017. . URL https://doi.org/10.1016/j.bdr.2016.10.001.

[15] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. *ACM International Conference Proceeding Series*, pages 121–124, 2013. .

[16] A. Delpeuch. Opentapioca: Lightweight entity linking for wikidata. *arXiv preprint arXiv:1904.09131*, 2019.

[17] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, E. Motta, and H. Sack. AI-KG: an automatically generated knowledge graph of artificial intelligence. In J. Z. Pan, V. A. M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, editors, *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2020. . URL https://doi.org/10.1007/978-3-030-62466-8_9.

[18] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, and E. Motta. Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. *Future Gener. Comput. Syst.*, 116:253–264, 2021. . URL https://doi.org/10.1016/j.future.2020.10.026.

[19] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van De Walle. RML: A generic language for integrated RDF mappings of heterogeneous data. In *CEUR Workshop Proceedings*, volume 1184, 2014.

[20] A. Dimou, T. De Nies, R. Verborgh, E. Mannens, and R. de Walle. Automated Metadata Generation for Linked Data Generation and Publishing Workflows. *Proceedings of the 9th Workshop on Linked Data on the Web*, 1593, 2016. ISSN 1613-0073.

[21] A. Dridi and D. R. Recupero. Leveraging semantics for sentiment polarity detection in social media. *Int. J. Mach. Learn. Cybern.*, 10(8): 2045–2055, 2019. . URL https://doi.org/10.1007/s13042-017-0727-z.

[22] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić. Introducing wikidata to the linked data web. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *The Semantic Web – ISWC 2014*, pages 50–65, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11964-9.

[23] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler. Knowledge Graphs. *Knowledge Graphs*, mar 2020. . URL http://arxiv.org/abs/2003.02320.

[24] O. Fodor and H. Werthner. Harmonise: A step toward an interoperable e-tourism marketplace. *International Journal of Electronic Commerce*, 9(2):11–39, 2005. . URL https://doi.org/10.1080/10864415.2005.11044324.

[25] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling Ontology Evaluation and Valilidation - in Proceedings of ESWC2006. *Eswc2006*, page 15, 2006.

[26] D. Gazzè, A. L. Duca, A. Marchetti, and M. Tesconi. An overview of the tourpedia linked dataset with a focus on relations discovery among places. *ACM International Conference Proceeding Series*, 16-17-Sept:157–160, 2015. .

[27] M. Grüninger, M. S. Fox, and M. Gruninger. Methodology for the design and evaluation of ontologies. In *International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing*, pages 1–10, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.8723.

[28] R. V. Guha, D. Brickley, and S. Macbeth. Schema.org: Evolution of Structured Data on the Web. *Communications of the ACM*, 59(2): 44–51, jan 2016. ISSN 0001-0782. . URL https://dl.acm.org/doi/10.1145/2844544.

[29] A. Halterman. Mordecai: Full Text Geoparsing and Event Geocoding. *The Journal of Open Source Software*, 2(9):91, 2017. ISSN 2475-9066. .

[30] A. Halterman. Mordecai: Full text geoparsing and event geocoding. *The Journal of Open Source Software*, 2(9), 2017. .

[31] M. Hepp. GoodRelations: An ontology for describing products and services offers on the web. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5268 LNAI, pages 329–346, 2008. ISBN 3540876952. .

[32] P. Heyvaert, B. De Meester, A. Dimou, and R. Verborgh. Declarative rules for linked data generation at your fingertips! *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11155 LNCS(August): 213–217, 2018. ISSN 16113349. .

[33] L. Iannone, A. Rector, and R. Stevens. Embedding Knowledge Patterns into OWL. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl, editors, *The Semantic Web: Research and Applications*, pages 218–232, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-02121-3.

[34] E. Kärle, U. Simsek, Z. Akbar, M. Hepp, and D. Fensel. Extending the schema.org vocabulary for more expressive accommodation annotations. In R. Schegg and B. Stangl, editors, *Information and Communication Technologies in Tourism 2017*, pages 31–41, Cham, 2017. Springer International Publishing. ISBN 978-3-319-51168-9.

[35] E. Kärle, U. Şimşek, O. Panasiuk, and D. Fensel. Building an ecosystem for the tyrolean tourism knowledge graph. *arXiv*, pages 260–267, 2018. ISSN 23318422. .

[36] K. I. Kotis, G. A. Vouros, and D. Spiliotopoulos. Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations. *The Knowledge Engineering Review*, 35:e4, 2020. .

[37] R. L R and T. Bomatpalli. A survey of travel recommender system. *International Journal of Computer Sciences and Engineering*, 7(3): 356–362, 2019. URL https://doi.org/10.26438/ijcse/v7i3.356362.

[38] J. B. Lamy. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80:11–28, 2017. ISSN 18732860. .

[39] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015. ISSN 22104968. .

[40] P. W. Lord. The semantic web takes wing: Programming ontologies with tawny-owl. *ArXiv*, abs/1303.0213, 2013.

[41] J. L. Martinez-Rodriguez, A. Hogan, and I. Lopez-Arevalo. Information extraction meets the semantic web: A survey. *Semantic Web Journal*, 11:255–335, 2020. .

[42] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems - I-Semantics '11*, pages 1–8, New York, New York, USA, 2011. ACM Press. ISBN 9781450306218. . URL http://dl.acm.org/citation.cfm?doid=2063518.2063519.

[43] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

[45] C. Möller, J. Lehmann, and R. Usbeck. Survey on english entity linking on wikidata. *arXiv preprint arXiv:2112.01989*, 2021.

[46] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 271–280, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312295. . URL https://doi.org/10.1145/2187836.2187874.

[47] N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*, page 25, 2001. ISSN 09333657. .

[48] A. Orme, H. Tao, and L. Etzkorn. Coupling metrics for ontology-based system. *IEEE Software*, 23(2):102–108, mar 2006. ISSN 0740-7459. . URL http://ieeexplore.ieee.org/document/1605186/.

[49] S. Ou, V. Pekar, C. Orasan, C. Spurk, and M. Negri. Development and alignment of a domain-specific ontology for question answering. *Proceedings of the 6th International Conference on Language Resources and Evaluation, LREC 2008*, pages 2221–2228, 2008.

[50] A. Sakor, K. Singh, A. Patel, and M.-E. Vidal. Falcon 2.0: An entity and relation linking tool over wikidata. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3141–3148, 2020.

[51] J. F. Sequeda, W. J. Briggs, D. P. Miranker, and W. P. Heideman. A pay-as-you-go methodology to design and build enterprise knowledge graphs from relational databases. In C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, editors, *The Semantic Web – ISWC 2019*, pages 526–545, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30796-7.

[52] N. Shuyo. Language detection library for java, 2010. URL http://code.google.com/p/language-detection/.

[53] M. G. Skjæveland, H. Forssell, J. W. Klüwer, D. P. Lupp, E. Thorstensen, and A. Waaler. Pattern-based ontology design and instantiation with reasonable ontology templates. In *WOP@ISWC*, 2017.

[54] S. Staab, C. Braun, I. Bruder, A. Düsterhöft, A. Heuer, M. Klettke, G. Neumann, B. Prager, J. Pretzel, H. P. Schnurr, R. Studer, H. Uszkoreit, and B. Wrenger. GETESS—searching the web exploiting German Texts. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1652, pages 113–124, 1999. ISBN 3540663258. . URL http://link.springer.com/10.1007/3-540-48414-0_7.

[55] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *16th International World Wide Web Conference, WWW2007*, pages 697–706, 2007. ISBN 1595936548. .

[56] G. Tamašauskaitė and P. Groth. Defining a knowledge graph development process through a systematic review. *ACM Trans. Softw. Eng. Methodol.*, feb 2022. ISSN 1049-331X. . URL https://doi.org/10.1145/3522586. Just Accepted.

[57] M. Tenemaza, J. Limaico, and S. Luján-Mora. Tourism recommender system based on natural language classifier. In T. Z. Ahram, W. Karwowski, and J. Kalra, editors, *Advances in Artificial Intelligence, Software and Systems Engineering*, pages 230–235, Cham, 2021. Springer International Publishing. ISBN 978-3-030-80624-8.

[58] R. Troncy, G. Rizzo, A. Jameson, O. Corcho, J. Plu, E. Palumbo, J. C. Ballesteros Hermida, A. Spirescu, K. D. Kuhn, C. Barbu, M. Rossi, I. Celino, R. Agarwal, C. Scanu, M. Valla, and T. Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46-47:2–13, 2017. ISSN 15708268. . URL http://dx.doi.org/10.1016/j.websem.2017.07.002.

[59] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org, 2016.

[60] R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, and R. Van de Walle. Querying datasets on the web with high availability. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *The Semantic Web – ISWC 2014*, pages 180–196, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11964-9.

[61] R. Verborgh, M. V. Sande, P. Colpaert, S. Coppens, E. Mannens, and R. Van De Walle. Web-scale querying through linked data fragments. In *CEUR Workshop Proceedings*, volume 1184, 2014.

[62] D. Xiao, N. Wang, J. Yu, C. Zhang, and J. Wu. A Practice of Tourism Knowledge Graph Construction Based on Heterogeneous Information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12522 LNAI(c):159–173, 2020. ISSN 16113349. .

[63] C. Zhang, H. Yao, C. Huang, M. Jiang, Z. Li, and N. V. Chawla. Few-shot knowledge graph completion. 2020.

[64] W. Zhang, H. Cao, F. Hao, L. Yang, M. Ahmad, and Y. Li. The Chinese Knowledge Graph on Domain-Tourism. *Lecture Notes in Electrical Engineering*, 590(November):20–27, 2020. ISSN 18761119. .

[65] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. . URL https://www.sciencedirect.com/science/article/pii/S2666651021000012.