UNICA

UNIVERSITÀ
DEGLI STUDI
DI CAGLIARI

Ph.D. DEGREE IN
Mathematics and Computer Science

Cycle XXXVI

TITLE OF THE Ph.D. THESIS

Knowledge Graphs and Large Language Models for Intelligent Applications in the Tourism Domain

Scientific Disciplinary Sector(s)

S.S.D. INF/01

Ph.D. Student:	Luca Secchi
Supervisor	Prof. Gianni fenu
Co-Supervisor	Prof. Diego Reforgiato Recupero

Final exam. Academic Year 2022/2023
Thesis defence: February 2024 Session

Abstract

In the current era of big data, the World Wide Web is transitioning from being merely a repository of content to a complex web of data. Two pivotal technologies underpinning this shift are Knowledge Graphs (KGs) and Data Lakes. Concurrently, Artificial Intelligence has emerged as a potent means to leverage data, creating knowledge and pioneering new tools across various sectors. Among these advancements, Large Language Models (LLM) stand out as transformative technologies in many domains.

This thesis delves into an integrative exploration, juxtaposing the structured world of KGs and the raw data reservoirs of Data Lakes, together with a focus on harnessing LLM to derive meaningful insights in the domain of tourism.

Starting with an exposition on the importance of KGs in the present digital milieu, the thesis delineates the creation and management of KGs that utilize entities and their relations to represent intricate data patterns within the tourism sector. In this context we introduce a semi-automatic methodology for generating a Tourism Knowledge Graph (TKG) and a novel Tourism Analytics Ontology (TAO). Through integrating information from enterprise data lakes with public knowledge graphs, the thesis illustrates the creation of a comprehensive semantic layer built upon the raw data, demonstrating versatility and scalability. Subsequently, we present an in-depth investigation into transformer-based language models, emphasizing their potential and limitations. Addressing the exigency for domain-specific knowledge enrichment, we conduct a methodical study on knowledge enhancement strategies for transformers based language models. The culmination of this thesis is the presentation of an innovative method that fuses large language models with domain-specific knowledge graphs, targeting the optimisation of hospitality offers. This approach integrates domain KGs with feature engineering, enriching data representation in LLMs.

Our scientific contributions span multiple dimensions: from devising methodologies for KG construction, especially in tourism, to the design and implementation of a novel ontology; from the analysis and comparison of techniques for enriching LLMs with specialized knowledge, to deploying such methods in a novel framework that effectively combines LLMs and KGs within the context of the tourism domain.

In our research, we explore the potential benefits and challenges arising from the integration of knowledge engineering and artificial intelligence, with a specific emphasis on the tourism sector. We believe our findings offer a promising avenue and serve as a foundational platform for subsequent studies and practical implementations for the academic community and the tourism industry alike.

Keywords: Knowledge Graphs, Ontology Design, Tourism Ontology, Semantic Data Lake, Knowledge Injection, Large Language Models, Transformers, BERT, Classification, Natural Language Processing, Feature Engineering, Information Extraction, Web Science, Web Mining, Tourism, Hospitality,

Acknowledgements

This thesis is the culmination of the last three years of my professional and personal life. The period of my Ph.D. studies largely coincided with the COVID pandemic on one hand, and with a very unique phase of my life on the other. I embarked on this journey at 48 and conclude it at 51, navigating a rocky and often challenging path that has, however, been full of stimuli and has paved new ways for my future. Engaging with academic research has not been easy and taught me the meaning of the word "uncertainty", which in this case means stepping out of the comfort zone where work outcomes are predictable and risks are calculable. In the world of research, nothing is certain by definition, and results might not materialize. In the end, the role of a researcher is a metaphor for the human condition: to live in the present day not knowing the future ahead.

Fortunately, I was not alone in this journey, during which I was accompanied by many individuals who have helped me stay on track. Among these, I want to primarily thank my supervisor, Prof. Gianni Fenu, who has been a consistent and reliable reference in every situation. At the same time, I wish to thank Prof. Diego Reforgiato Recupero and colleagues from the Media Institute of Open University, Francesco Osborne and Angelo Salatino, who have supported and advised me at many junctures throughout the Ph.D. journey. A special thanks to Alessandro Chessa who believed in me, and to Vincenzo de Leo and Andrea Cadeddu with whom I've shared many discussions and reflections over the years.

Lastly, I want to express my gratitude to my family for enduring and supporting me in this slightly crazy endeavor. First and foremost, my wife Laura, without whom I would have been lost long ago, and then my children, Nicola and Emma, who patiently indulged a father who wanted to return to being a student like them.

Publications

The research reported in this thesis has contribuited to the following publications:

- [1] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo Salatino, and Luca Secchi. **Enriching Data Lakes with Knowledge Graphs.** Presented at 1st International Workshop on Knowledge Graph Generation From Text co-located with 19th Extended Semantic Conference (ESWC 2022) - CEUR Workshop Proceedings, 3184:123–131, 2022.

- [2] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo Salatino, and Luca Secchi. **Data-driven methodology for knowledge graph generation within the tourism domain.** IEEE Access, 11:67567–67599, 2023.

- [3] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. **Optimizing tourism accommodation offers by integrating language models and knowledge graph technologies.** 2023. Submitted

- [4] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. **Leveraging Knowledge Graphs with Large Language Models for Classification Tasks in the Tourism Domain.** 2023. Presented at Workshop DEEP LEARNING FOR KNOWLEDGE GRAPHS co-located with the 22ND INTERNATIONAL SEMANTIC WEB CONFERENCE (ISCW 2023) - CEUR Workshop Proceedings Vol. 3559, article no. 4.

- [5] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. **A comparative analysis of knowledge injection strategies for large language models in the scholarly domain.** 2023. Submitted

- [6] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. **Enhancing Scholarly Understanding: A Comparison of Knowledge Injection Strategies in Large Language Models**, 2023. Presented at Workshop DEEP LEARNING FOR KNOWLEDGE GRAPHS co-located with the 22ND INTERNATIONAL SEMANTIC WEB CONFERENCE (ISCW 2023) - CEUR Workshop Proceedings Vol. 3559, article no. 7.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Research Contributions

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 [7]. The World Wide Web Consortium (W3C) has developed and promoted different standards, like RDF/S [8], OWL [9] and SPARQL [10], 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 [11]. A few examples of knowledge graphs publicly available are DBpedia [7], YAGO (Yet Another Great Ontology) [12] or WikiData [13]. 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 knowledge graph are modelled according to a domain ontology, which gives meaning to the represented information and supports inferring new knowledge. Thanks to these characteristics many organizations are building Enterprise Knowledge Graphs to support their business.

Following an alternative path, many organisations are developing data analytics and intelligent applications adopting a very different approach based on data lakes [14]. Data lakes are repositories of data stored in natural/raw format. A data lake does not impose a schema on the data when it's written, which may include structured data from relational databases, semi-structured data (i.e., JSON, CSV), unstructured data (i.e., text data), or binary data (i.e., images, audio, video). It is usually built on top of cost-efficient infrastructures such as Hadoop, Amazon S3, MongoDB, ElasticSearch, etc. Several organisations rely on data lakes for crucial tasks such as data analytics or machine learning tasks. A major limitation of this solution is that without descriptive metadata and a mechanism to maintain it, such data tend to be noisy, making their management and analysis complex and time-consuming.

By leveraging data lakes and knowledge graphs we are not limiting ourselves to structured and quantitative data (typically contained in Relational Databases and Data Warehouses), but we are dealing with all kinds of data. More specifically, we are referring here to how information can be extracted from text data by using Natural Language Processing and Language Models (LM) in particular [15, 16]. These models have proven state of the art performances in many different tasks such as entity recognition, text classification, sentiment analysis, etc. It is still an open research theme how domain-specific knowledge can be used to improve LM performances and in particular how knowledge graphs can be used to this end.

In this thesis, we have investigated the use of Knowledge Graphs and data lake technologies powered by NLP algorithms in the Tourism domain through new methodologies. Moreover, the Tourism Analytics Ontology is presented and used to build a new tourism knowledge graph for London and Sardinia. Subsequently, the combination of Knowledge Graphs and Language Models is examined with a comparative study of different knowledge injection strategies, with an evaluation in the scholarly domain. Finally a new system that combines knowledge graphs and language models to support revenue management in the hospitality field is introduced. The thesis adds value to the scientific literature through its findings and available resources.

The research program undertaken during the Ph.D. course focused on investigating and developing methodologies for constructing knowledge graphs and ontologies on top of data lake repositories. Additionally, it explored the integration of knowledge graphs and ontologies with Language Models to facilitate the creation of intelligent applications in the tourism domain.

The main research questions addressed are:

Q1. how can we guide the design of a new ontology with a data driven approach?

Q2. how can we build a knowledge graph starting from a data lake repository to improve its value?

Q3. what methods are best suited to enhance Language Models with information from knowledge graphs in order to improve specific tasks like classification?

Q4. how can we leverage knowledge graphs together with Language Models to build intelligent applications?

By answering to these research questions, the contributions provided in this research work are:

- a general data-driven methodology for the semi-automatic generation of knowledge graphs, which we have applied to the tourism domain;

- the new Tourism Analytics Ontology (TAO);

- an open-source program for producing the Tourism Analytics Ontology (TAO) using code and data;
- an instance of the tourism knowledge graph (TKG) containing data related to two tourist destinations (Greater London and Sardinia island in Italy);
- an open-source pipeline for generating a tourism knowledge graph from (semi-)structured and unstructured data stored in a data lake;
- an evaluation assessing functional, logical, and structural dimensions of TAO and TKG;
- a comparative analysis of different strategies for enhancing Language Models with knowledge graphs;
- a new benchmark composed of 15K research articles for the evaluation of knowledge enhancement methods;
- a novel methodology that effectively integrates language models and knowledge graphs in the context of the tourism domain;
- the application of the proposed approach to support revenue management in the tourism domain;
- a comprehensive evaluation demonstrating the advantages of the proposed solution compared to conventional transformer models;
- an in-depth analysis of feature engineering techniques to identify the most effective combination of features when combining knowledge graphs and language models.

1.2 Dissertation Structure

The thesis is organized as follows:

- Chapter 2 provides an introduction to knowledge graphs, language models, and data lakes, which form the foundational concepts upon which this research study is built.
- Chapter 3 analyzes the background of current research on Knowledge Graphs construction and presents a data driven methodology for Knowledge Graph generation from data lakes within the tourism domain. This chapter stems from the collective efforts made in conjunction with Professor Gianni Fenu and Professor Diego Reforgiato, both from the *University of Cagliari*, along with other esteemed researchers from *Linkalab S.r.l.* in Italy, and the *Knowledge Media Institute of The Open University* located in Milton Keynes, United

Kingdom. The outcomes of these efforts have been documented in publications [1, 3].

- Chapter 4 provides an introduction to the current research regarding the combination of knowledge graphs and language models and presents a comparative analysis of different strategies enhancing Language Models with knowledge graphs. This chapter is a product of a joint endeavour with Professor Gianni Fenu and Professor Diego Reforgiato from the *University of Cagliari*, as well as other distinguished scholars from *Linkalab S.r.l.* in Italy and the *Knowledge Media Institute of The Open University* in Milton Keynes, United Kingdom. The results of this collaborative work have been reported in publications [5, 6].

- Chapter 5 presents a novel approach employed to enhance revenue management strategies within the hospitality industry in the realm of tourism. This chapter is a product of a joint endeavour with Professor Gianni Fenu and Professor Diego Reforgiato from the *University of Cagliari*, as well as other distinguished scholars from *Linkalab S.r.l.* in Italy and the *Knowledge Media Institute of The Open University* in Milton Keynes, United Kingdom. The results of this collaborative work have been reported in publications [3, 4].

- Finally, Chapter 6 offers a detailed analysis of our research outcomes and explores potential future directions in this field.

Chapter 2

Knowledge graphs, data lakes and language models.

In this chapter, we introduce some fundamental concepts to which we refer throughout the remainder of this thesis. These concepts are related to knowledge graphs (Section 2.1), data lakes (Section 2.2 and Large Language Models (Section 2.3).

2.1 Knowledge graphs

The concept of a ”knowledge graph” is subject to debate, with various definitions proposed [17, 18]. These definitions range from specific technical suggestions to broader, more inclusive proposals. In this context, we adopt an inclusive definition, proposed by [11], that considers *a knowledge graph as a graph of data designed to accumulate and convey knowledge about the real world.* The graph consists of nodes representing entities of interest and edges representing relationships between these entities. The underlying data graph follows a graph-based data model, which could be a directed edge-labelled graph or a property graph, among other alternatives.

Knowledge, in this context, refers to established facts or information. This information can either be derived externally or extracted from the knowledge graph itself. Knowledge can range from straightforward declarations like ”Cagliari is a city in Italy” to quantified statements like ”all cities are places.” While basic statements can be represented as edges in the data graph, accumulation of quantified statements necessitates a more comprehensive format, such as ontologies. This allows for deduction, enabling additional knowledge creation, for example, concluding that ”Cagliari is a place.” Furthermore, the knowledge graph can utilize inductive methods to garner more knowledge based on both simple and generalized statements.

Knowledge graphs, in essence, serve as dynamic stores of shared information within specific entities or organizations [19]. They come in two main forms: open and enterprise knowledge graphs. Open knowledge graphs, such as DBpedia [7], Freebase [20], Wikidata [21], and YAGO [12], are publicly accessible and cover a wide range of domains thanks to various sources like Wikipedia or community-driven initiatives. On the flip side, enterprise knowledge graphs are used internally for business purposes, seen across industries like web search, commerce, and social networks, with notable examples like Google [22], Amazon [23], and Facebook [19].

2.1.1 Data model

In this work we applied a *Directed edge-labelled graphs* data model [24] to implement the proposed Knowledge Graphs presented in Chapters 3 and 5. A directed edge-labelled graph is a set of nodes and a set of directed labelled edges between these nodes. In this kind of knowledge graphs, nodes represent entities while edges illustrate (one to one) relations between these entities. In this context, the addition of information usually involves adding new nodes and edges. If the information is incomplete, it is handled by simply not including a specific edge. This graph-based data representation provides a flexible way to incorporate new data sources compared to conventional relational models, which require predefined schemas.

Unlike other structured data models like trees (XML, JSON, etc.), graphs do not impose a hierarchical organization of data and permit the representation and querying of cycles. The Resource Description Framework (RDF), recommended by the W3C [8], is a standardized data model based on directed edge-labelled graphs. It defines different types of nodes, including Internationalized Resource Identifiers (IRIs) [25] for web entity identification, literals for scalar values representation (e.g. strings or numbers), and blank nodes, which are anonymous nodes without an identifier. RDF represents data as a graph using triples. These triples are composed of a subject, a predicate, and an object, and they collectively express a relationship or a fact. *Subject*, is the resource or entity the triple describes. It's usually represented as an IRI, but it can also be a blank node. *Predicate*, is the property or relationship type that connects the subject to the object. It is represented as an IRI. *Object*, is the value or target of the relationship. It can be an IRI (referring to another resource), a blank node, or a literal value (such as a string or number). The set of triples forms a graph, where each triple represents an edge connecting two nodes. The subject and object are nodes in the graph, while the predicate represents the edge or link.

We applied in Chapter 3 a specific data model based on the so called *graph dataset*. Rather than merging multiple directed edge-labelled graphs into a single large graph, a graph dataset, composed of multiple named graphs and a default graph, facilitates more efficient data management, especially when dealing with data from various sources. The flexibility of this arrangement allows for easier data updates, source reliability assessment, and data integration across graphs. This model is particularly significant in managing and querying interlinked Web-based RDF graphs where the tracking of data origin becomes crucial [26].

2.1.2 Graph queries

In Chapter 3 we have implemented Knowledge Graphs queries using SPARQL [10]. SPARQL, an acronym for SPARQL Protocol and RDF Query Language, is a prominent querying language and protocol designed for the extraction and manipulation of data stored in Resource Description Framework (RDF) format, used as the foundation of knowledge graphs. Developed by the World Wide Web Consortium (W3C), SPARQL is widely acknowledged for its robustness, adaptability, and the ability to perform complex queries on RDF datasets [27].

The syntax of SPARQL enables queries to match patterns within the RDF graph, with options for conjunctions, disjunctions, and source restrictions [10]. The language supports multiple forms of queries including select, construct, describe, and ask, offering versatility in the return of results as either tabular formats, RDF graphs, or simple boolean values.

SPARQL also provides for federated querying across different RDF graphs, a feature that is of particular relevance given the distributed nature of many RDF datasets on the web [28] and that was directly leveraged in Chapter 3. The language's filter function allows for the reduction of result sets based on specified criteria, while its solution modifier permits ordering, projection, distinct and reduced solutions, and limiting and offsetting result sets.

2.1.3 KG schema

Semantic schema offer a structure to define high-level terms used within a graph, aiding in reasoning about these terms. In essence, semantic schemas are about defining sets of nodes based on the types of entities they denote. For instance, we could create classes like 'Accommodation', 'Tourist Destination', etc., for groupings. We can then establish connections or relations between these classes. Another application of semantic schema is the definition of the semantics of edge labels, or properties. We might infer that certain properties are sub-properties of a broader property. For instance, 'bus' and 'flight' could be sub-properties of 'connects to'. Like classes, properties can also form a hierarchy. We may also want to define the domain of properties, indicating the classes of entities from which edges with that property start. Similarly, the range of properties can be defined, pointing to the class(es) of entities towards which edges with that property lead.

Notably, the RDF Schema (RDFS) [29] is a widely recognized standard for defining a semantic schema for (RDF) graphs. It allows for defining subclasses, subproperties, domains, and ranges among classes and properties used in an RDF graph. Such definitions can also be serialized as a graph. Furthermore, the semantics of terms used in a graph can be defined in greater detail and richness using the Web Ontology Language (OWL) [9] standard for RDF graphs. We applied these semantic schema languages when creating TAO in Section 3.4.3.

Semantic schema often come into play with incomplete graph data where the absence of an edge, such as a connection between two cities, doesn't imply that such a relationship doesn't exist in reality. This contrasts with the Closed World Assumption (CWA) often used in traditional database systems, which posits that the data completely describes the world, thus any absent relationships are considered non-existent. The alternative, the Open World Assumption (OWA), suggests that an absent edge does not make a relationship false. As such, the introduction of new edges under CWA might contradict previously held assumptions (because of a missing information considered as negation statement), but under OWA, a false statement remains false even with additional data. In the context of large-scale, diverse, and incomplete data represented by graphs, the Open World Assumption (OWA) serves as the most suitable default semantics. However, there could be scenarios where we want to ascertain a certain level of "completeness" in our data graph or its specific sections. For example, we may want all accommodations to have a name, an address, a number of rooms and beds, so applications can rely on this minimum information. In such cases, we can establish constraints in a validating schema to ensure the existing graph's data validity. This differs from semantic schema, which are used to infer new graph data. Shapes, a standard method for creating validating schema for graphs, target a set of nodes in a data graph and specify constraints for them. Validating schema and semantic schema, although serving different purposes, can supplement each other. Two prevalent shape languages for RDF graphs, Shape Expressions (ShEx) [30] and SHACL (Shapes Constraint Language) [31], are used for validating graphs in various domains, including healthcare, scientific literature, and spatial data. This concepts are very important when validating an ontology and a knowledge graph as we do in Section 3.6.

2.1.4 Entities disambiguation and labelling

Because knowledge graph's nodes represent real entities it is always possible that two or more such entities have the same name. To mitigate ambiguity, we can implement two strategies: first, use globally unique identifiers to prevent naming conflicts when incorporating external data into the knowledge graph, and second, utilize external identity links to disambiguate nodes by referencing to an external source. We can create persistent identifiers (PIDs) to uniquely denote an entity and avoid conflicts, notable examples of which include DOIs for papers, ORCID iDs for authors or ISBNs for books, among others. In the Semantic Web context, the RDF data model recommends using global Web identifiers for nodes and edge labels. It proposes using Internationalised Resource Identifiers (IRIs) [25] instead of Uniform Resource Locators (URLs) for identifying non-information resources such as cities or events[1]. When such an HTTP IRI is queried, the server can return a

[1]Uniform Resource Identifiers (URIs) come in two types: Uniform Resource Locators (URLs), which are utilized for locating informational resources, and Uniform Resource Names (URNs), which are employed for naming non-informational resources. Additionally, Internationalised Resource Identifiers (IRIs) are a category of URIs that support Unicode characters.

description of that entity, potentially in RDF format, facilitating linkage of RDF graphs to external entities. However, HTTP IRIs are not necessarily persistent. To overcome this, Persistent URL (PURL) services offer redirection from a central server to a particular location, ensuring persistence of identifiers even when the document location changes. An entity's identity can be grounded by associating it with uniquely identifying information, or by using identity links to state equivalence to a co-referent entity in an external source, with the owl:sameAs property in the OWL standard exemplifying this concept.

Entity identifiers may be globally unique yet not always human-readable. For instance, Wikidata uses codes like wd:Q84 for London and code wd:Q1462 for Sardinia, which provides longevity and avoids language bias. To humanize such identifiers, it's common to append labels using triples like `wd:Q1462 rdfs:label "Sardinia"`, providing linguistic reference information. Such labels facilitate cross-referencing with text corpora and help users connect nodes to real-world entities. Additional context can be provided via aliases, using triples such as `wd:Q1462 skos:altLabel "Sardigna"`, or comments like `wd:Q2887 rdfs:comment "Sardinia is the second-largest island in the Mediterranean Sea."`. Literal nodes, like ”Sardinia”, can be paired with a language code for multilingual representation. For example, RDF allows to use triples like `wd:Q1462 rdfs:label "Sardinia"@en` or `wd:Q1462 rdfs:label "Sardegna"@it` to indicate labels in English and Italian. Knowledge graphs that include human-readable labels, aliases, and comments in various languages are often referred to as multilingual lexicalised knowledge graphs.

We used globally unique identifiers, links to external sources, multilingual labels and comments when building the Tourism Knowledge Graph in Chapter 3.

2.1.5 Representing data

Graph data models permit nodes to be defined as datatype values. For example, RDF employs XML Schema Datatypes (XSD), among others, to represent such datatype nodes as a pair (l,i), where 'l' is a lexical string and 'i' is an IRI that signifies the datatype, e.g., *“2023-07-17T10:00:00”^^xsd:dateTime*. These nodes, called literals in RDF, are disallowed from having outgoing edges. Common datatypes include xsd:string, xsd:integer, xsd:decimal, and xsd:boolean, with xsd:string as the default when the datatype is omitted. RDF-based applications can then interpret these datatypes, perform operations like equality checks, ordering, or casting, in accordance with their standard definitions.

2.1.6 Ontologies

Ontologies in computing provide a precise, formal representation of the meaning of terms within a particular scope, like a domain. Their effectiveness relies on the agreement on the ontology's definitions, its detail level, and its adoption breadth and consistent application. The more an ontology is agreed upon and adopted across various knowledge graphs, the more interoperable those knowledge graphs become. The default ontology language for RDF graphs is the Web Ontology Language (OWL).

Interpreting a data graph involves mapping its nodes and edges to another graph (the domain graph) made up of real-world entities connected by real-world relations. This interpretation process is essentially creating a bridge between data terms and domain terms. Different interpretations may abide by different assumptions. For instance, the Unique Name Assumption (UNA) disallows interpretations that map two data terms to the same domain term, while the Non-Unique Name Assumption (NUNA) allows them. Under the Closed-World Assumption (CWA), a domain graph can only satisfy a data graph if it entails the same edges. In contrast, the Open World Assumption (OWA) allows a domain graph to satisfy a data graph without entailing the same edges, as long as it doesn't contradict any edge.

OWL adopts NUNA and OWA, making it a more general case where multiple nodes/edge-labels can refer to the same entity/relation-type (NUNA), and anything not entailed by the data graph isn't assumed to be false (OWA). In addition to these base assumptions, semantic conditions can be added to certain patterns in the data graph to define which interpretations satisfy it. These semantic conditions form the ontology language features.

When examining an ontology it is possible to identify a TBox (Terminological Box) and an ABox (Assertional Box). The TBox contains the terminological knowledge of the domain, which is more about the schema or the structure. It defines the classes, properties, and their relationships, representing the "conceptual schema" of the domain. In the context of OWL, the TBox contains class, property, and datatype definitions, along with associated constraints and characteristics. On the other hand ABox contains the assertional knowledge, which refers to specific instances of classes and properties defined in the TBox. It provides facts about the world based on the schema provided in the TBox. For example, if the TBox defines "Person" and "hasAge" properties, the ABox might contain data like "John is a Person and hasAge 30." In OWL, ABox statements are often the individual assertions, linking specific instances to classes and other instances using properties.

OWL provides various features to describe individual entities. First, binary relations between individuals can be asserted using edges. Additionally, OWL allows defining relations that explicitly state two terms refer to the same or different entities. Negation, indicating that a relation doesn't hold, can also be used and serialized as a graph using reification[2]. Further features of OWL include defining subproperties, domains, and ranges for properties, but also additional aspects. Properties can be defined as equivalent, inverses, or disjoint. A property can denote transitive, symmetric, asymmetric, reflexive, or irreflexive relations. Multiplicity of the relation

[2]Reification is a mechanism in RDF used to represent statements about statements, allowing for the expression of metadata about individual triples. It involves creating a new resource that represents the original statement and using standard RDF properties to describe that resource [32].

denoted by properties can be defined as being functional (many-to-one) or inverse-functional (one-to-many). Also, a key can be defined for a class, representing the set of properties uniquely identifying the entities of that class. Lastly, a property can be related to a chain, meaning pairs of entities related by the chain are also related by the given property. OWL provides several features for defining classes. First, it allows defining two classes as equivalent or disjoint. Then, OWL allows creation of new classes through set operators on existing classes or based on conditions that the properties of its instances meet. New classes can be defined as a complement of another class, the union or intersection of other classes, or an enumeration of all instances. Alternatively, by placing restrictions on a property 'p', classes can be defined where instances are entities that have: a value from a given class on 'p'; an individual as a specific value on 'p'; themselves as a reflexive value on 'p'; at least, at most or exactly a certain number of values on 'p' (cardinality); and at least, at most or exactly some number of values on 'p' from a given class (qualified cardinality).

We used many OWL features when creating TAO ontology in Section 3.4.3 and perform the ontology evaluation taking into account the same features in Section 3.6.

2.2 Data Lakes

The last decades have seen a data production explosion, largely due to the rapid development of mobile devices, Internet of Things (IoT) and social media. While this vast amount of data presents immense opportunities, its volume, velocity, and varied sources and structures challenge traditional data management systems [33]. Data warehousing has been a longstanding solution for managing and processing large amounts of data. However, while these systems are effective for structured data, they struggle with semi-structured and unstructured data, which constitute a majority of big data [33].

To address these challenges, the concept of a data lake was introduced [34]. The term "data lake," coined by James Dixon in 2010 [14], initially referred to a raw, single-source data storage system, alternative to traditional data marts [35]. However, current definitions encompass data from numerous sources [36, 37, 38] and include preprocessed data and analytical results storage [39]. Financial viability for data lakes necessitates affordable storage. Although often associated with Hadoop[3] and the HDFS[4] [14, 36], data lakes represent a concept, with Hadoop being just one potential enabling technology among many [40]. A varied tool landscape for data

[3]Hadoop is an open-source framework for distributed storage and processing of large datasets across clusters of computers using simple programming models. Developed by the Apache Software Foundation, Hadoop's core components are the Hadoop Distributed File System (HDFS) for storage and the MapReduce computational model for parallel processing

[4]HDFS (Hadoop Distributed File System) is the primary storage system of Hadoop, designed to store very large data sets reliably across clusters of machines while providing high throughput access to this data. HDFS splits large files into blocks, distributes them across nodes in a cluster, and replicates each block multiple times to ensure fault tolerance and data durability.

lakes is presented in [37, 40], where the best tool to manage and process specific data is employed. This complex and articulated landscape has been further expanded by the widespread affirmation of cloud computing and the emergence of proprietary cloud solutions from big players in the field like Amazon Web Services, Microsoft Azure or Google Cloud [40].

Data lake architecture organizes data within a data lake to facilitate specific use cases. Two principal variants exist: zone and pond architectures. Zone architectures [41, 42], despite their differences in the number and characteristics of zones, share a common principle: data are assigned to a zone based on the degree of processing applied to them. Data are initially stored in raw format in the raw zone, and further conditioned in other zones for various purposes, such as standardization, cleansing, or preparation for specific use cases. One advantage is that data, even if pre-processed, can still be accessed in their raw form. In contrast, the pond architecture [43] distributes data across five different ponds, each catering to specific data types: raw, analog, application, textual, and archival. Unlike zone architectures, data exist in only one pond at any given time, and as they move, they undergo transformations specific to each pond. While this allows easy analysis of pre-processed data, it contradicts the data lake concept as data lose their original format once they leave the raw data pond.

Because in a data lake, data are preserved in their raw state until required for analysis, only at this point, a schema is implemented to enable examination, a practice known as "schema on read" [44]. Although this approach ensures data are untouched until use, it is not always the most efficient for performance and cost optimization. Therefore, sometimes data are transformed and stored in certain file formats like Parquet, AVRO, or ORC that can also manage schema information. When integrating data from various sources, contexts, and schemas, metadata become essential to maintain an understanding of these data. This holds true for data lakes, where metadata management is a key component [39, 41, 43]. Metadata provide information about the actual data, such as schema, semantics, or lineage, thereby facilitating the discovery, reliability, and usability of data. There are numerous approaches to manage metadata, with data catalogs often utilized in data lake literature for metadata storage [45]. Whenever data are incorporated into the data lake, related metadata must be added to the catalog, allowing users to search the catalog for additional information like schema, relationships, or provenance. However, these catalogs do not encompass all metadata pertinent to data lakes and no comprehensive metadata management strategy covering all data lake metadata is available [34].

To avoid the transformation of data lakes into "data swamps"— places where data becomes inaccessible, unwieldy, costly, and useless— metadata management is crucial. This ensures the data remains usable and reachable. To alleviate this risk, the utilization of a semantic layer can be highly beneficial. This semantic layer, using formal ontologies and semantic technologies, provides a cohesive, unified representation of data throughout the organization. In Chapter 3 we propose a methodology

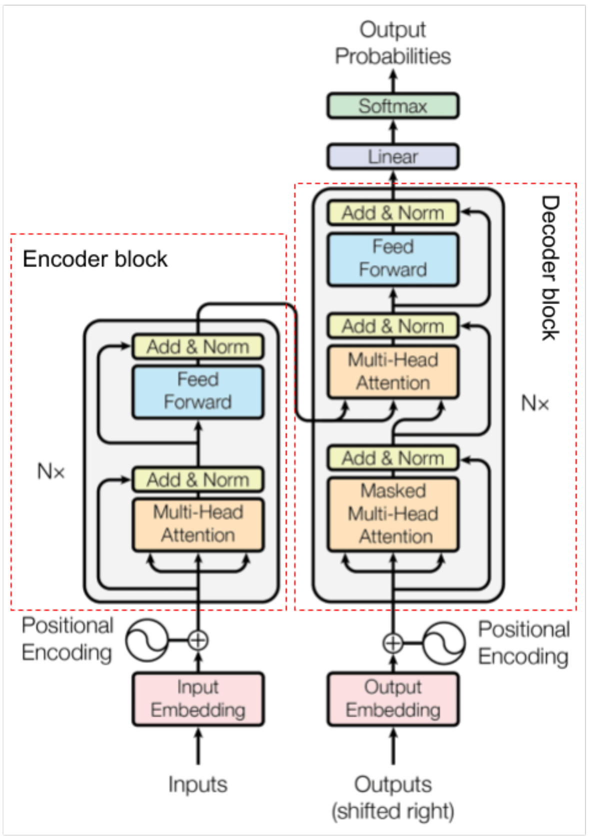


Figure 2.1: Transformer architecture with Encoder and Decoder blocks highlighted in red.

to extend a data lake containing data extracted from touristic platforms with a semantic layer produced as a knowledge graph.

2.3 Transformer Based Language Models

Early Natual Language Processing (NLP) systems were predominantly rule-based and later supplanted by machine learning models. While effective, these models require custom feature engineering, which demands domain expertise and can be time-consuming. However, with the advent of advanced computer hardware and word embeddings[5] such as Word2Vec [46] and Glove [47], the use of deep learning models like CNNs [48] and RNNs [49, 50] in NLP systems increased.

[5]Word Embeddings are a type of word representation that captures the semantic meaning of words by mapping them to vectors of real numbers in a predefined vector space. We discuss embeddings in Section 2.3.3

A first limitation of traditional deep learning models, such as CNN and RNN, is that they struggle with modeling long-term contexts and learn a word representations aggregating information from its neighbours losing long range relations between words (locality bias) [51]. Furthermore, RNNs, due to their sequential processing, limit the utilization of parallel computer hardware. To overcome these shortcomings, Vaswani et al. [15] proposed the Transformer model, a deep learning model based on self-attention. This model, consisting of a stack of encoder and decoder layers, allows for better parallelization compared to RNNs and can easily take into account long-term contexts. Other important challenges of deep learning models are that they requires a significant number of labeled instances, which are costly and laborious to generate, and that they must be trained from scratch for each downstream task, which has high computation costs. To overcome the need for large training datasets a learning paradigm called Self Supervised Learning was introduced [52, 53]. On the other hand, transfer learning, where a model trained for a source task is repurposed and adapted for a similar target task, was leveraged to addresses the need for training each model from scratch [52].

Inspired by the success of pretrained CNN models in Computer Vision and leveraging the use of Transformer based architectures, the NLP research community developed Transformer-based pretrained language models (T-PTLMs), combining the power of transformers and self-supervised learning [54].

2.3.1 Transformer based Architectural variants

While he original Transformer architecture, proposed by [15] and shown in Figure 2.1, consists of one stack of encoder layers and one of decoder layers, initial T-PTLMs, were developed based on either transformer decoder (GPT) [55, 56, 57] or encoder layers (BERT) [16]. Encoder-based T-PTLMs, like BERT and RoBERTa [58], consist of an embedding layer followed by multiple encoder layers, with the output from the last layer serving as the final contextual representation of the input sequence. These models are typically used in Natural Language Understanding (NLU) tasks. Decoder-based T-PTLMs, such as GPT models, comprise an embedding layer followed by a stack of decoder layers, which include only masked multi-head attention plus feed-forward network layers and where multi-head module for encoder-decoder cross attention is not present. These models are generally used in Natural Language Generation (NLG) tasks. Encoder-decoder based T-PTLMs, like MASS [59] and BART [60], are ideal for sequence-to-sequence tasks like Machine Translation and Text Summarization.

All these models support transfer learning, which can be adapted to different tasks through fine-tuning or prompt-tuning on target datasets. We used BERT model in Chapters 4 and 5 in combination with Knowledge Graphs.

2.3.2 Self-supervised Learning

Self-supervised learning (SSL) is an emerging learning paradigm in the Artificial Intelligence (AI) research field. Its appeal lies in its ability to utilize unlabeled data to infuse foundation knowledge about language [52, 53], image [61, 62], or speech [63, 64] into pre-trained models.

The importance of SSL arises from the limitations of supervised learning, which has been a significant driver of AI progress. While supervised learning models, trained on human-annotated instances, excel at specific tasks, they require a large volume of labeled instances for optimal performance, but the process of data collection and labeling is both time-consuming and costly. Moreover these models often suffer from generalization error as they learn only from the training data available. SSL, on the other hand, does not require human-labeled data and enhances the model's generalization ability by learning from large volumes of unlabeled data. The drawbacks of supervised learning, such as the fundamental reliance on human-labeled datasets and low generalization capability, make SSL an attractive alternative. In SSL the supervision is provided by the same data through the definition of specific tasks. This is different from supervised learning, where humans provide the labels for the data, and unsupervised learning, where the model learns patterns in the data without any label. An example of a self-supervised learning task in the contest of NLP would be predicting the next word in a sentence. The model is given part of a sentence and learns to predict the next word based on the context provided by the preceding words. This approach allows the model to learn the structure and semantics of the language without needing explicit labels, instead the model learns universal language representations by solving these pretraining tasks over a vast amount of unlabeled data. These representations, encoding both syntax and semantic information, are useful in downstream tasks where they help the model to achieve better performance using fewer labeled instances.

2.3.3 Embeddings in T-PTLMs

Deep learning models, including transformers, require numerical input, necessitating the mapping of input text to a sequence of dense, low-dimensional vectors, known as embeddings in natural language processing. Transformer-based pretrained language models favor character or sub-word embeddings [65, 66] over word embeddings due to their smaller vocabulary size and ability to represent any word, thus addressing the out-of-vocabulary (OOV) word issue common with word embeddings while still being able to encode fine-grained information in word representation. Besides representing input data, embeddings are also used to encode additional information like position [58, 67, 68], language [69], etc. We used tokenizers[6] and sub-word em-

[6]Tokenizers are algorithms or tools that break down input text into smaller units, called tokens, suitable for processing by transformer-based language models. These tokenizers handle various linguistic structures, manage subword units like wordpieces or byte-pair encodings, and ensure

beddings in Chapters 4 and 5 to process input texts for the proposed classification model.

2.3.4 Pretraining language models

As described above, the standard approach for language models involves pretraining on large volumes of unlabeled text and then fine-tuning on small task-specific datasets.

In the realm of natural language processing, researchers have developed a vast number of pre-trained models like BERT [16], RoBERTa [58], ELECTRA [68], XLNet [70], and T5 [71] offering numerous advantages to other research practitioners. After the model is pre-trained with SSL it acquires universal language representations as described in Section 2.3.2. By merely adding a small number of task-specific layers, pre-trained models can then be tailored to downstream tasks, eliminating the need for training the entire model from scratch and offering a valuable starting point. Moreover, pretraining enhances the model's performance in downstream tasks even with limited datasets, thereby decreasing the need for a vast amount of labeled instances. Given that deep learning models, due to their high parameter count, are prone to overfitting on smaller datasets, pretraining serves as a form of regularization by providing a suitable initialization and preventing overfitting. Pretraining a model involves five key steps [54]:

- Preparation of the Pretraining Corpus: this involves obtaining and cleaning unlabelled text from one or more sources. Research has shown that using a larger text corpus from multiple sources can improve model performance [58, 70, 71]. Deduplication of the corpus can also lead to similar performance with fewer training steps [72].

- Generation of the Vocabulary: transformer-based pretrained language models typically use tokenizers like WordPiece [65], Byte Pair [66] and SentencePiece [73] to generate the vocabulary. The vocabulary usually consists of all unique characters and sub-words or words more frequently used. Different models use different tokenizers and generate vocabularies of varying sizes.

- Design of the Pretraining Tasks: the model learns language representations by minimizing losses based on one or more pretraining tasks. These tasks should pose a challenge hard enough to enable the model to learn semantics at different levels (word, sentence or document) [58, 74], provide high density of training information to limit the training corpus size [68], and be similar to downstream tasks [75].

- Choice of the Pretraining Method: instead of just using SSL for pretraining, other hybrid methods like Knowledge Intensive PreTraining (KIPT) [76, 77],

that the tokenized input aligns with the model's pre-trained vocabulary.

which combines SSL and Knowledge Distillation (KD)[7], can be applied to reduce the computation costs and the time required to complete the pretraining phase. Moreover, methods like continual pretraining with new vocabulary [79, 80] or adapt and distill [81] can be used for adapting general models to specific domains.

- Choice of the Pretraining Dynamics: it has been shown by [58] that it is possible to enhance the model performance by choosing key hyperparameters and training data size when designing the pretraining process, like applying large batch sizes or adding more pretraining steps.

It is worth noting that pre-training a model is a practice that requires sizeable textual data and expensive computation resources. When it is not possible to satisfy this requirements it is necessary to pursue different approaches more suitable in a low-budget / low-computation resource setting, as those we examine in Chapter 4 and apply in Chapter 5.

Many pretrained models are public available fore research or commercial purposes. Huggingface website offers a vast catalog of such models[8]. In Chapters 4 and 5 we used one pretrained model for BERT[9] to support our research.

2.3.5 Fine-tuning language models

As described in Section 2.3.4, a language model can be provided with a general understanding of language through pretraining, but task-specific knowledge is required for the model to excel in specific downstream tasks. This implies that the model's weights need to be optimized for the target task. Task-specific knowledge is introduced via fine-tuning, where the model's weights are adjusted based on the task-specific loss [67]. Furthermore, fine-tuning establishes a wide separation between different labels thus improving the model performance [82]. It is worth noting that, although fine-tuning updates all transformer layers, including the embedding layer, the higher layers (those nearer to the output) undergo more significant changes compared to the lower ones [82, 83, 84, 85].

We applied fine-tuning to BERT based models in Chapters 4 and 5 where we used knowledge enhancement technique to improve the proposed downstream tasks without requiring a new pretraining phase.

[7]Knowledge Distillation is a compression method where a student model learns from a teacher model how to generalize by reproducing its behaviour. This allows the student model to reach similar performances compared to the teacher model. This method was introduced by [78].

[8]See `https://huggingface.co/models`

[9]See `https://huggingface.co/bert-base-uncased`

2.3.6 BERT for text classification

BERT (Bidirectional Encoder Representations from Transformers) is a notable language model capable of capturing extensive contextual representations of words and sentences [67] based on Transformers technology. Text classification is one of the primary applications of BERT [86]. In this thesis we used BERT to classify texts in Chaper 4 and Chapter 5.

BERT can be fine-tuned on task-specific labeled data to effectively learn the prediction of accurate class labels for new, unseen text data. The bidirectional characteristic of BERT allows it to understand the context of a word from both words coming before and after, leading to a more profound understanding of textual dependencies and relationships. BERT has become a benchmark model for text classification [87], demonstrating remarkable performance enhancements and the capability to manage intricate linguistic patterns and contexts. The BERT-base model is composed of an encoder with 12 Transformer blocks, each containing 12 self-attention heads and a hidden size of 768. Before BERT can process a text, it needs to be divided into individual tokens. These tokens can symbolize words, subwords, or even characters, based on the chosen tokenization approach. BERT can handle input sequences of up to 512 sub-words tokens to produce a representation of the sequence. In terms of text classification, BERT uses the final hidden state `(h)` of the initial token `[CLS]` as the representation for the entire sequence. To compute the probability `(p)` of a label `(c)`, a pooling layer is incorporated with a fully connected layer with a task-specific parameter matrix `(W)`. The output from the pooling layer is then input to an output layer that produces the output probabilities for each class. Depending on the nature of the classification task (binary vs. multi-class), different activation functions may be used. For binary classification, a sigmoid activation function is common. For multi-class classification, a softmax activation function is typically used to produce probabilities for each class. In case of multi-class classification the formula would be:

$$p(c|h) = softmax(Wh)$$

In case of binary classification the formula would be:

$$p(c|h) = sigmoid(Wh)$$

The parameters of BERT and W are co-tuned during fine-tuning.

Chapter 3

A Knowledge Graph for Tourism

With the tourism and hospitality industries gaining increased relevance in recent times, businesses in these sectors are constantly striving to offer innovative services. Concurrently, (big-) data has become this age's "new oil", with Knowledge Graphs and data lakes surfacing as the ideal methods to accumulate, refine, and organize disparate information. In this chapter, we introduce a strategy for semi-automatically creating a Tourism Knowledge Graph (TKG), which can facilitate a range of intelligent services within the tourism industry, and a novel ontology for representing this sector, named the Tourism Analytics Ontology (TAO). Our method involves processing and combining data stored in an enterprise data lake, supplemented with public knowledge graphs (DBpedia and GeoNames). The resulting knowledge graph ultimately serves as a semantic layer for the initial data lake. Owing to its modular design the Knowledge Graph can be conveniently expanded to encompass additional data sources or to implement new enhancement and refinement functions. We present an in-depth evaluation of the functional, logical, and structural aspects of TKG and TAO.

3.1 Introduction

The tourism industry is a prime candidate for the application of knowledge graph technologies as well as data lakes we introduced in Chapter 2, given the need of participants in this sector to combine large amount of data from a variety of disparate sources. This integration allows for a comprehensive representation of tourist destinations and all relevant parties involved [88, 89, 90].

A tourist destination is generally the primary factor in a tourist's decision to travel[1] and is typically defined by two elements: supply and demand. The supply aspect pertains to the capacity and readiness of providers to produce goods and

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

services and bring them to market. Comprehending the supply side of tourism covers all areas connected to tourism products and attractions (such as accommodations, events, points of interest, restaurants, and so on). Conversely, demand pertains to the type and amount (quantity) of a product or service that consumers desire. Understanding the factors that affect the demand side of tourism covers all areas related to tourists' preferences and point of view or their attributes (such as socio-demographic, classification, origin). This information is vital for shaping business and marketing strategies as well as backing a range of software and services in this sector, like search engines and recommendation systems [91, 92].

The application of big data to scrutinize various facets of tourism is gaining importance in research as demonstrated by [93], this is where data lakes become essential due to their capacity to facilitate big data storage and analysis using machine learning and AI. However, data lakes necessitate governance to ensure continuous upkeep and to maintain the data usability and accessibility. To address this challenge, it is beneficial to depend on a semantic layer: a data representation based on semantic technologies and a formal ontology that can provide a unified, consolidated view of data across the organisation. There have been multiple attempts to provide a semantic layer to data lakes, each targeting a specific domain of application [94, 95, 96, 97, 98, 99, 100, 101]. Nevertheless, as far as we know, no one has yet applied this solution in the tourism industry.

In general, developing KGs is an expensive and time-consuming endeavor, even with the aid of mapping languages like RML [102, 103, 104]. In particular, it remains a challenge to automatically generate KGs from multiple semi-structured and textual sources (for example, descriptions of specific accommodations, reviews, etc.) to depict the many aspects of the tourism sector, such as the different types of accommodations and amenities. As a result, many KGs in this area are no longer maintained [89, 90] or cannot be easily expanded to other tourist destinations [103]. Moreover, the relevant ontologies, such as Accommodation Ontology[2], Schema.org[3], and Hontology [105], are to some extent incompatible with each other (as discussed in section 3.4.3) and do not provide a detailed representation of some key entities (like amenities).

In this chapter, we present a universal, reproducible, and easily expandable methodology for KG generation and the resulting framework for semi-automatically creating a *Tourism Knowledge Graph (TKG)*, which amalgamates information from an enterprise data lake enhanced with public knowledge graphs (DBpedia and GeoNames). The resulting knowledge graph is ultimately used as a semantic layer for the original data lake. This comprehensive characterisation of tourism can be used to facilitate the quantitative analyses of a tourist destination and support several intelligent services. To model this data, we developed the Tourism Analytics Ontology (TAO), which provides a more detailed characterisation of tourist locations, lodging

[2]`http://ontologies.sti-innsbruck.at/acco/ns.html`
[3]`https://schema.org/docs/hotels.html`

facilities, and amenities than previous solutions and can be effortlessly reused by similar initiatives.

We demonstrate our solution by applying it to Sardinia and London tourist destinations, generating over 10M triples that describe nearly 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]. Lastly, for the sake of reproducibility, we share the code base for our knowledge graph generation pipeline, for engineering TAO, and the evaluation tests[6].

In summary, the contributions of this chapter are as follows:

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

The remainder of this chapter is structured as follows. Section 3.2 discusses related works about semantic layers for data lakes, different knowledge graphs within the tourism industry and methodologies for their creation. Section 3.3 introduces the methodology adopted to guide the knowledge graph creation, Section 3.4 details the first three iterative phases related to the use cases refinement and ontology design. Section 3.5 describes the other three phases of the adopted methodology related to the creation of the proposed knowledge graph.

Section 3.6 presents the evaluation, and finally, Section 3.7 concludes the chapter with conclusions and future directions of work.

3.2 Related work

This section will encompass a review of previous literature focused on tree primary themes relevant to this work: i) the methods for ontology and knowledge graph

[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`

construction, ii) the extension of a data lake with a semantic layer and iii) the use of knowledge graphs in the field of tourism.

3.2.1 Ontology and Knowledge Graph generation

The process of constructing, preserving, and enhancing knowledge graphs necessitates the implementation of various ontology engineering methodologies (OEMs). Three categories of such methodologies have been classified by Kotis et al. [106]: collaborative, non-collaborative, and custom. A collaborative OEM is well-structured and involves the participation of knowledge engineers and domain experts in all stages of ontology development. A non-collaborative OEM, while not emphasizing stakeholder collaboration, still provides a clear and formal definition of stages, tasks, and workflows. A custom OEM may not formally define these elements but seeks to involve communities of practice and use tools to develop ontologies in an agile and decentralized manner, often collaboratively.

Several works in literature have explored the generation of knowledge graphs and their methodologies, each within its unique constraints and domains [107, 108, 109, 110]. The challenges encountered depend on these constraints which the developers need to meet. For instance, the construction of knowledge graphs from complex database schemas presents certain challenges (primarily related to size) that need to be addressed, such as efficient table reading, column consideration, and linked table mapping. Sequeda et al. [107] proposed an innovative approach to tackle the complexities of understanding database schemas, demonstrating its application in a large corporation. Tamašauskaitė and Groth [108] conducted a systematic review of knowledge graph creation processes, outlining steps such as data identification, knowledge graph ontology construction, knowledge extraction, extracted knowledge analysis, knowledge graph creation, and maintenance. This last step involves periodic updates and edits to the existing knowledge graph. The authors provide guidelines, best practices, and tools for the creation and maintenance of knowledge graphs.

In this chapter, we introduce a data-driven method that combines the semi-automatic generation of the knowledge graph using various existing tools and the construction of a supportive domain ontology using a collaborative OEM (as defined in [106]). This method is used to create a Knowledge Graph within the tourism sector.

3.2.2 Semantic layers for data lakes

Dibowski et al. [97] explored how to enhance the findability, accessibility, interoperability, and reusability of data stored in a data lake. They highlighted the advantages of supporting data lakes with ontologies and knowledge graphs, including data cataloguing, provenance tracking, access control, and semantic search. They developed the DCPAC ontology in relation to vehicle-produced data management.

Similarly, Diamantini et al. [95] presented a semantic model for the effective utilization of data stored in a data lake. They created a knowledge graph to assist in the correct identification of data by mapping the dimensions of analysis, indicators of interest, and formulas. Pomp et al. [101] faced similar issues regarding the collection, location, understanding, and access of vast data sources with the aim of providing real-time availability. They established a data lake to streamline the process from data collection to analysis, and proposed a semantic data platform called ESKAPE for the semantic annotation of ingested data. Additionally, they designed a knowledge graph that serves as an evolving index in accordance with the included data, allowing users to conveniently identify and analyze data from various sources. Bagozi et al. [94] suggested a semantics-based approach for personalized exploration of data lakes within the context of smart cities. They initially equipped the data lake with a semantic model using domain ontologies, then adopted an additional ontology to describe indicators and analysis dimensions. Finally, they generated personalized exploration graphs for different user types. Lastly, Ansari et al. [96] proposed a semantic profiling tool for metadata extension in data lake systems with the objective of understanding data meaning. The tool uses domain vocabularies and ontologies to recognize data meaning at schema and instance levels.

Differently to the methods mentioned above, we suggest a methodology that enriches a data lake containing data extracted from tourist platforms with a semantic layer implemented as a knowledge graph to expose data directly in RDF.

3.2.3 Knowledge Graphs in the Tourism Sector

In recent times, numerous attempts have been undertaken to construct knowledge bases across various areas, including tourism, leveraging data collected from online sources and social media platforms.

For example DBtravel [90] is a tourism-focused knowledge graph derived from the community travel site Wikitravel. It leverages the guidelines provided by Wikitravel for contributors and extracts the named entities available in the Spanish version of Wikitravel[7] using an NLP pipeline. Unfortunately, the knowledge graph and the source code used to produce it are no longer maintained or available online.

The 3cixty platform [88], constructed during Expo Milano 2015, aimed to build extensive knowledge bases encompassing descriptions of events and activities, landmarks and attractions, transit facilities, and social activities, collated from a plethora of both local and global sources, including hyper-local resources. Utilizing this model platform, new instances were created for London, Madeira, and Singapore, as well as the French Riviera during the years 2016-2017. However, the project appears to have been abandoned with no source code made available for the recreation of the infrastructure. While a SPARQL endpoint is still operational, it only permits data export in HTML, not RDF.

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

The Tourpedia platform, envisioned as the DBpedia of tourism, was created as part of the OpeNER Project [89]. OpeNER (Open Polarity Enhanced Name Entity Recognition) was a project funded under the 7th Framework Program of the European Commission with the primary objective of implementing a pipeline for natural language processing. Despite the project no longer being maintained, anyone can run the suggested pipeline to view categories, place information, and create and manage events and tour plans for users. Additionally, on the main website, it is still possible to run the web demo application, which displays sentiment about places through an interactive map. Some datasets are still available for download, but other tools, including the SPARQL endpoint, are no longer functional.

Other projects show that semantic technologies and knowledge graphs can be effectively applied to the field of tourism when data is extracted from curated proprietary sources. In the instance of *La Rioja Turismo* Knowledge Graph, Alonso-Maturana et al. [103] collect and integrate information pertaining to attractions, accommodations, tourist routes, activities, events, restaurants, and wineries from varied and diverse management systems. This approach focuses on the La Rioja Turismo ecosystem and cannot be readily extended to other tourist destinations. A similar situation applies to the Tyrolean Tourism Knowledge Graph [111], that leverages data gathered from destination management organisations (DMOs) and their IT service providers. In this instance, the creation of the knowledge graph relies on the availability of Schema.org annotations in the source websites, which was made possible through the cooperation of Tyrolean DMOs. Once more, this scenario is not always feasible as it requires a central organization to coordinate the disparate stakeholders.

Additional cutting-edge solutions include the creation of a knowledge graph by extracting information from the existing encyclopedia knowledge graph and unstructured web pages in Chinese [102, 112]. Besides the focus on the Chinese language, the authors concentrated on semi-structured knowledge extraction and deep learning algorithms to extract high-level entities and relations from unstructured travel notes. The project is not currently maintained and did not provide a SPARQL endpoint.

All the presented examples in literature demonstrate that the automatic generation of a tourism-focused knowledge graph that integrates key data sources in this domain and can be readily extended to other tourist destinations still poses a significant challenge. We also lack a single ontology[8] capable of providing a detailed 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 chapter aims to bridge this gap by introducing the Tourism Analytics Ontology (TAO)[9] In addition, we provide a SPARQL endpoint and plan to regularly update our knowledge graph using the pro-

[8]We analyze other ontologies in Section 3.4.3.

[9]We detail TAO in Section 3.4.3 and Appendix C, which offers a detailed characterization of accommodations, tourist locations and destinations[10], and a universal, reproducible, and easily

posed pipeline. Unlike the approaches discussed earlier, our proposal can be readily reused and extended to different tourist destinations since we release the complete source code, enabling other users to generate new KGs from multiple data sources that offer global coverage.

3.3 Methodology Overview

The methodology we employ for building the Knowledge Graph (KG) aligns with the general methodology discussed in [108]. The process is divided into six primary phases that can be performed repeatedly to refine the KG outcome. The first three phases form the core of a **data-driven design procedure** which extracts knowledge from the data sources to guide the refinement of use cases and ontology creation. The final three phases lead the actual creation and distribution of the knowledge graph together with its validation. The various phases are outlined in Figure 3.1.

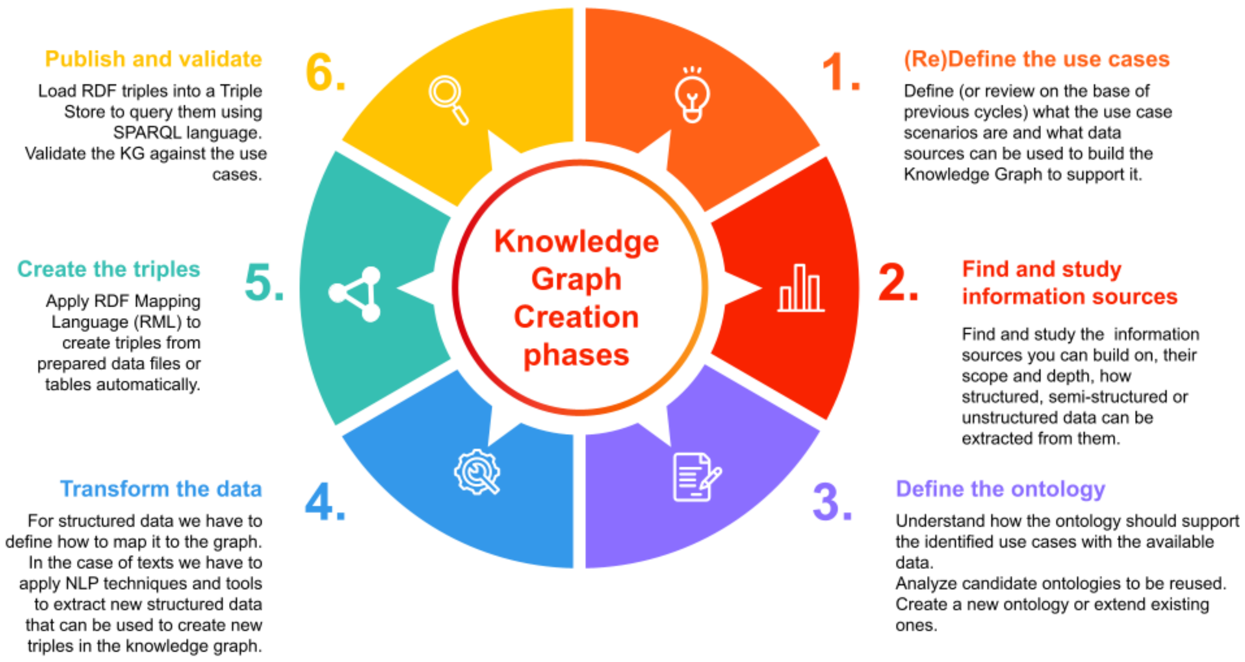

Knowledge Graph Creation phases
1.
(Re)Define the use cases
Define (or review on the base of previous cycles) what the use case scenarios are and what data sources can be used to build the Knowledge Graph to support it.
2.
Find and study information sources
Find and study the information sources you 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 and validate
Load RDF triples into a Triple Store to query them using SPARQL language.
Validate the KG against the use cases.

Figure 3.1: Tourism Knowledge Graph generation phases.

The *initial phase* concentrates on defining the use cases that the knowledge graph will support, essentially, the intended results a user or an application should be able to derive from it. As our process is data-driven, this is a preliminary definition that

extendable pipeline to integrate relevant data sources and generate a knowledge graph for the tourism sector.

is open to further refinement and should be reviewed multiple times until all use cases are effectively supported by the KG.

The *second phase* involves comprehending how the available data can aid the use cases, and also extracting knowledge from the data to assist in defining the ontology. The data is used to adjust the use cases to the actual information at our disposal, hence expanding the scope for some use cases or reducing it for others. Furthermore, the data is scrutinized to guide the ontology creation process.

The *third phase* is centered on the development of an ontology to model lodging, tourist destinations, and locations that support all the use cases outlined in the first phase and supports the domain knowledge identified in the second phase.

The *fourth phase* involves transforming the data gathered from the data sources to ready it for triple creation in the subsequent phase. During this process, semi-structured data is processed using a variety of data wrangling approaches, while unstructured texts are treated using natural language processing.

The *fifth phase* deals with triple creation using the data transformed in the previous phase. The RDF Mapping Language (RML) is used for triple creation to also include the transformation process metadata in the knowledge graph.

Lastly, the *sixth phase* is dedicated to the validation of the knowledge graph against the use cases and to its publication in a triple store.

The proposed methodology is versatile and can be used whenever there is a need to design a KG and its supporting ontology from the ground up. It is particularly useful for modelling a KG to support applications that are constrained by available data sources that impose specific limits on the design and implementation process. In the Section 3.4, we describe in detail the first three phases related to use case refinement and ontology creation. In Section 3.5 we will describe the final three phases, related to the generation of the knowledge graph, encompassing its implementation, publishing and validation.

3.4 Design phases

The following subsections provide a comprehensive description of the initial three phases which pertain to use case development, investigation of information sources, and the development of ontology. Further details about the last three phases, associated with the generation of the knowledge graph, will be discussed in Section 3.5.

3.4.1 Define the use cases

We commence with an initial broad definition of some of the use cases that we aim to address during the construction of the KG, while also considering potential data sources that could aid them. We should also determine the sorts of applications we would need to develop on top of the KG to assist the use cases. This examination can provide us a broader perspective of how the KG could be utilized. This is important in understanding how well the data sources can accommodate the scenario and guide the design process in shaping the KG. Indeed, this phase is interconnected with the second phase (i.e., *Find and study information sources*), discussed in Section 3.4.2, because we need to take into account the information that can be leverged to support the chosen use cases. It is also linked to the third phase (i.e., *Define the ontology*) in Section 3.4.3, because we can adopt different design strategies for the KG generation depending on the types of methods and applications it should assist (e.g., do we need to apply reasoning techniques to the KG?). In our case, to create a KG that can aid the analysis of tourist destinations in terms of supply and demand, we have identified the following use cases in collaboration with domain experts and stakeholders:

(**UC1**) Assist in identifying the topics that tourists discuss in their reviews;

(**UC2**) Aid in identifying the topics of interest presented in the descriptions of lodging facilities[11] and accommodation[12] offers;

(**UC3**) Aid in recognizing and linking tourism entities in the KG for various applications in the domain of social media, news, and blogs;

(**UC4**) Assist in sentiment analysis [113, 114] applications about tourists' attitudes toward lodging facilities and destinations;

(**UC5**) Assist in classifying tourist destinations based on their offerings and on tourist opinions.

(**UC6**) Support the optimization of the touristic offer composition and proposal strategies.

We also identified several applications that can utilize the KG to yield superior results (see [115] for a comprehensive overview of applications based on knowledge graphs). In turn, each of the following applications can be used to better assist one or more use cases:

1. **automatic reasoning**[13] and **graph learning**[14] on the KG allows for the inference of new triples thus enriching the explicit knowledge other applications can work on; for this reason, it is indirectly related to all use cases;

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

[12]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.)

[13]Utilizing Description Logic and OWL.

[14]Using Graph Neural Networks or analogous techniques.

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 enhancing 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 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. **Classification** of entities like accommodations or places can support use cases 1, 2, 5 and 6.

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 crucial to understand that, the actual feasibility of a use case can be confirmed only when the knowledge graph is constructed and one or more of the supporting applications are implemented. In Chapter 5 we have developed one such application that can be used to optimize accommodation offers by leveraging the knowledge graph described here.

3.4.2 Find and study information sources

The use cases outlined in section 3.4.1 necessitate the identification of a basic range of information sources that can support the development of an initial *core* version of the Tourist Knowledge Graph. Once this primary Knowledge Graph is established, it can be expanded with additional information sources following the same methodology laid out in this chapter. The flexibility and extensibility of knowledge graphs make this approach feasible.

This study builds on the outcomes of an industry project on Tourism 4.0 named *Data Lake Turismo*, carried out by Linkalab s.r.l.[15], which evolved from a prior research initiative backed by the Digital Innovation Hub of Sardinia[16] and Fondazione Banco di Sardegna[17]. The structure and technologies of the data lake are detailed

[15]Linkalab s.r.l. is a small Italian firm specializing in data science and data engineering. Website `https://www.linkalab.it/`

[16]`https://www.dihsardegna.eu/`

[17]`https://www.fondazionedisardegna.it/`

in Appendix A. The project's goal was to develop a digital platform for analysing tourism data. A key feature of this platform was a data lake designed for the collection, transformation, and analysis of data in the tourism sector. Despite its coverage, the project lacked a semantic layer that could enhance and support data analysis. This limitation worked as the inspiration and starting point for the work presented here. Using the data lake infrastructure, we can access data related to accommodation facilities, user reviews, and opinions; these data are then enhanced with DBPedia and Geonames.

Based on the use cases, the following information are needed:

- Details about lodging facilities and their offerings;
- User opinions expressed by their reviews;
- information about tourist locations (i.e. points of interest such as a railway station or a lake);
- information about tourist destinations.

Beginning with what was initially available in the data lake, we identified a preliminary set of information sources that were augmented with additional ones to sufficiently cover the required information scope:

- Booking.com[18], a digital travel firm specializing in hotels, B&Bs, and other types of accommodation; its website provides information about accommodations, related offers, as well as user reviews.
- Airbnb[19], a US based company that links hosts offering their spaces (like apartments, rooms, etc.) with travellers in need of a place to stay; its peer-to-peer model, which is derived from the sharing economy, represents a new emerging trend in the tourism and accommodation sector; its website provides information about accommodations, related offers, as well as user reviews.
- DBpedia[20], an open knowledge graph built with structured content extracted from various Wikimedia projects (e.g., Wikipedia). In particular, we link entities in TKG to the DBpedia entities of selected classes (e.g., `DBpedia:Places` or `DBpedia:Food`).
- GeoNames[21], a geographic database accessible through APIs and as RDFs documents. We link entities in TKG with GeoNames entities representing places.

[18]This source was already integrated into the data lake.

[19]This source was already integrated into the data lake.

[20]This source was not included in the data lake but it is a public knowledge graph accessible at `https://www.dbpedia.org/`

[21]This source was not included in the data lake but is a public knowledge graph accessible at `http://www.geonames.org/`

It should be noted that, despite the existence of numerous other websites and apps for tourism and hospitality, Booking.com and Airbnb are industry leaders and collectively cover both the traditional accommodation industry and the emerging sharing economy. A similar argument could be made for DBpedia and GeoNames when it comes to places (DBpedia and GeoNames) or general tourism-related topics (DBpedia).

The choices about data sources have implications for both the use case definition and the ontology development phases. While it might have been feasible to incorporate new data sources from the outset, doing so comes with associated costs. Where possible, it should be avoided, as our initial goal is to establish a core version of the knowledge graph before expanding its scope. On the other hand, it is crucial to always choose data sources that incorporate a comprehensive and well-established model of the relevant business sector (in this case, tourism) within the data itself. This supports the design of the ontology with a data-driven analysis process.

Exploration of Source Data

If we want to recognizing what type of data is available for use, reviewing the documentation is crucial, but it's also important to conduct an exploratory data analysis on the files and tables found in the source data lake to thoroughly understand its contents. In our case, the analysis was focalized on a subset of all resources found in the data lake:

- hospitality-related data:
  - details pertaining to hospitality establishments (such as hotels, bed and breakfasts, resorts) and their attributes (like name, address, category, hospitality amenities);
  - data concerning accommodations provided by a hospitality establishment (like a hotel room, B&B room, apartment);
  - rental propositions for accommodations (such as cost, capacity, etc.).
- user feedback data (like date, score, review).

Data is retrieved from the data lake in tables with nested structures and must be "flattened" for subsequent tasks to use. This is a result of the redundant and non-normalised way the data lake stores information.

The outcome of the exploratory analysis revealed:

- the organization of data in fields and sub-structures;
- the availability of structured and unstructured data (i.e., texts);
- the existence of texts in various languages without always specifying the language used;

- the presence of numbers, Boolean values, time/date values, or categorical values in structured data fields;
- the possibility of data being untyped and internally represented as strings;
- the lack of a connection between categorical data and a lookup table or taxonomy;
- the absence of unique IDs in some instances for resource identification.

This analysis guided us to establish some key data pre-processing steps to be carried out before constructing the knowledge graph and the ontology:

- **data preparation**: This step involved drawing out data from the source data lake using SQL queries; subsequently, we stored it on a local file system to be prepared (cleaned, flattened, combined) for subsequent tasks.
- **data enrichment**: During this step, we enhanced the data using various methods; specifically, we used NLP techniques to detect the language of the text (e.g., English, Italian, French, and so on), as it is essential for subsequent tasks to function correctly.

We also discovered that the data lake source needed to be supplemented with information about attractions and points of interest from other sources. To fulfill this requirement, we recognized DBpedia and GeoNames as the most suitable data sources for the following reasons:

i) both sources are well-established and regularly updated, with a large supportive community;

ii) both sources extensively cover the identified destinations (and many others);

iii) both sources are available as linked open data and APIs.

3.4.3 Define the ontology

To facilitate the use cases and associated applications we aim to produce, a KG that encompasses all pertinent entities and their relations. Therefore, we must establish a domain ontology capable of modelling these elements. To direct the construction of the ontology, we collaborated with domain experts from Linkalab to formulate both functional and non-functional requirements. These requirements were also articulated in a more practical manner through competency questions, in other words, queries articulated in everyday language [116, 117]. We provide a comprehensive description of the resulting competency questions and both types of requirements in Appendix B.

Competency Questions (CQ) are beneficial because they:

i) are easily comprehensible to individuals without technical expertise;

ii) serve as a concrete reference for the ontology construction process, indicating what needs to be accomplished;

iii) are readily testable over the course of the validation process.

We employed a data-driven design process and utilized two complementary strategies in the formulation of the competency questions:

i) top-down, by generating new questions with a domain specialist and then assessing whether our data could provide answers; and

ii) bottom-up, by extracting them from the information present in the original data.

Given that CQs rephrase and cover all functional requirements, we were able to confirm at the conclusion of this process that the ontology would adequately represent the data within the knowledge graph, thus fulfilling the use cases and bolstering the associated applications. Upon completion of this process, we pinpointed the key elements to represent within our domain ontology: i) lodging facilities (structures), ii) accommodations within these facilities, iii) amenities provided to tourists, iv) tourist destinations and sites, and v) user feedback. Subsequently, we scrutinised several cutting-edge ontologies in the tourism field (elaborated in Section 3.4.3), but none fully met our needs. As a result, we developed and implemented a novel ontology, the Tourism Analytics Ontology (TAO), taking advantage of existing ontologies (for instance, Schema.org, Hontology).

In the subsequent subsections, we will discuss:

i) the ontologies that served as our foundation; and

ii) the final iteration of TAO and the decisions behind its design.

Leveraging pre-existing ontologies

Multiple tourism-related ontologies were examined to determine their potential for reuse in our scenarios.

We discovered three primary categories of ontologies:

1. ontologies that rely on OpenTravel[22] or other robust industry standards, typically designed for information sharing among tourism entities (e.g., the Harmonise Ontology [118]).

[22]OpenTravel Alliance (OTA) is a not-for-profit trade association founded in the late 1990s that aims to develop open standards for the electronic exchange of business information within the travel industry. Their standards facilitate the distribution of travel products and services across different companies and platforms, ensuring interoperability between various systems in the travel, tourism, and hospitality sector.

2. ontologies developed by scholars to facilitate specific tasks, such as question answering (e.g., QALL-ME Ontology [119]) and information retrieval (e.g., GETESS [120]), as well as ontologies that integrate or build upon them (e.g., cDOTT [121], Hontology [105]).

3. ontologies that draw on Schema.org [122] and GoodRelations [123], like the Accommodation Ontology.

After considering the functional and non-functional requirements, we chose three of them: (i) the Accommodation Ontology, (ii) the Schema.org markup for hotels, and (iii) Hontology. The last one is currently not accessible as OWL serialisation at any particular IRI and does not appear to be actively maintained. TAO also integrates two other ontologies: (iv) GeoNames[23], used to identify geographic locations, and (v) the DBpedia ontology[24], used for further classification of locations and food varieties (e.g., pizza, sushi). In the following section, we will discuss the chosen ontologies and vocabularies and their application in TAO.

Accommodation Ontology (prefix `acco:`) is a derivative of GoodRelations[25] (prefix `gr:`) that focuses on detailing accommodation offers from an e-commerce viewpoint. It introduces additional vocabulary elements for describing hotel rooms, hotels, camping sites, and other accommodation types as well as their characteristics. However, it does not distinguish between the lodging facility (e.g., a hotel as a whole), and the individual accommodations available for lease (e.g., the hotel rooms), as all types of lodging facilities and accommodations are subclasses of the same class (`acco:Accommodation`). The Accommodation Ontology does not specify types of amenities (referred to as accommodation features) but instead "provides a consolidated conceptual model for encoding proprietary feature information". So, rather than providing classes for room and hotel features, the ontology defines the general class `acco:AccommodationFeature` that can contain feature information with varying degrees of formality. A lease offer is modelled using the GoodRelations property `gr:Offering`, specifying that the offer is a `gr:LeaseOut` via the property `gr:hasBusinessFunction`. Regrettably, the Accommodation ontology does not address several concepts that our use cases require, such as 1) tourist destinations (e.g., London), 2) tourist locations (e.g., lake, museum, train station), 3) tourist reviews.

Schema.org[26] markup for hotels (prefix `schema:`), incorporates and expands on many Accommodation Ontology [124] concepts, and models hospitality according

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

[24]`https://www.dbpedia.org/resources/ontology/`

[25]GoodRelations Ontology is designed for representing product, price, and company data on the web, particularly in the context of e-commerce. Developed by Martin Hepp, GoodRelations has been widely recognized and used for annotating offerings, business entities, and related information in a standardized way to improve the visibility of products and services in online search.

[26]Schema.org is a collaborative, community-driven project with support from major search engines such as Google, Bing and Yahoo!. Its goal is to create, maintain, and promote schemas for structured data on the Internet, web pages, emails, and beyond.

to three primary classes[27]:

1. A **lodging business**, (e.g., a hotel, hostel, resort, or a camping site): it represents at the same time the lodging facility, which is the hospitality physical structure and the governing business organisation. A lodging business can include multiple buildings but is usually a single location.

2. An **accommodation**, i.e., the specific units of the establishment (e.g., hotel rooms, apartments, camping pitches, etc.). These are the actual items that are offered for rental.

3. An **offer** to rent an accommodation, for a certain price and a specific usage (e.g., occupancy), potentially constrained by specific terms and conditions.

In this approach, there's a clear demarcation between the lodging business and the accommodation, due to the presence of two separate classes: `schema:Accommodation` and `schema:LodgingBusiness`. Regrettably, Schema.org is not designed to function as an OWL ontology because its data model is overly generic and stems from RDF Schema[28]. The primary objective of Schema.org is to facilitate the sharing of structured data on the Internet, while OWL relies on formal semantics that allow reasoning on the knowledge graph. Furthermore, the `schema:LodgingBusiness` class is not compatible with the GoodRelations ontology without causing logical inconsistencies. Specifically, Schema.org designates `schema:LodgingBusiness` as a subclass of `schema:LocalBusiness`, which is a subclass of both `schema:Organisation` and `schema:Place`. Conversely, GoodRelations asserts that `schema:Organization` and `schema:Place` are disjoint. We incorporated Schema.org into TAO by importing and expanding a few classes and properties, including `schema:PostalAddress`, `schema:UserReview`, `schema:address`, `schema:subjectOf`. We also chose suitable Schema.org types that describe places to enhance TAO tourism location classes, utilizing `rdfs:seeAlso` to create a mapping with them[29].

Hontology (prefix `ho:`) is a multilingual ontology designed for the accommodation sector (H signifies hotel, hostal, and hostel). It's a freely accessible domain-specific ontology in four languages: English, Portuguese, Spanish, and French [105, 125]. It was partially aligned with QALL-ME and Schema.org and presents several useful concepts in this domain such as Facilities (also known as amenities), Services, Staff, and Points Of Interest. The ontology isn't published as linked data but can be downloaded and used in a local environment. Its most recent version is from 2012, meaning it isn't aligned with the latest extensions of Schema.org. Moreover, as it's not based on GoodRelations, it doesn't meet our

[27]Commonly referred to as types in Schema.org.

[28]Refer to `https://schema.org/docs/datamodel.html`

[29]In this regard, we can view the TAO ontology as an external extension of Schema.org, as explained in the page `https://schema.org/docs/extension.html`

non-functional requirements. In TAO, we re-implemented some of its classes that describe location amenities, like `ho:Balance`, `ho:AirConditioning`, `ho:Ballroom`, and `ho:BeautySalon`.

DBpedia Ontology[30] (prefix `dbpedia:`) is a shallow, cross-domain ontology, manually crafted based on the most widely used infoboxes within Wikipedia[31]. The ontology currently encompasses 685 classes that form a subsumption hierarchy and are described by 2,795 distinct properties. We utilized some of these classes to enhance TAO tourist location types (subclasses of `tao:TouristLocation`) which were also mapped to GeoNames geographic features.

GeoNames Ontology[32] (prefix `gn:`) is designed to describe geographical features, specifically those defined in the geonames.org database. It has three main ontology classes: Feature (a set of all geospatial instances in GeoNames like cities and countries), Class (a set of all feature schemas defined in GeoNames), and Code (a set of abbreviated feature codes in different feature schemas). GeoNames Feature is utilized for modelling tangible geospatial entities (UK, Washington, Colosseum, etc.), while GeoNames Class and Code are employed for representing Features instances' meta-information. All feature instances are uniquely identified by IRI in GeoNames.

We used the GeoNames `gn:Feature` class to model lodging facilities and tourist locations, which are physical places, and to represent their mutual geographic relations using `gn:parentFeature`. We also used GeoNames to enrich TAO tourist location types with specific codes, for instance, `tao:Park` was linked to the `gn:L.PRK` code.

The Tourism Analytics Ontology Explanation

This section is dedicated to the explanation of the newly developed Tourism Analytics Ontology (TAO) and the discussion of the decisions made during its design. Our aim in creating this ontology was to ensure i) it meets all the requirements outlined in the Appendix B, ii) it has the capability to represent all pertinent information from the selected sources, and iii) it maintains full compatibility with the Accommodation Ontology, GoodRelations, and Schema.org. In particular, the Accommodation Ontology is directly imported using `owl:imports`, GoodRelations is indirectly imported via the Accommodation Ontology, and Schema.org is partially included by reusing specific classes and properties or creating explicit mappings to it.

The new ontology possesses the following features:

1. it includes the LodgingFacility class, which is representative of hotels, motels,

[30]`https://www.dbpedia.org/resources/ontology/`

[31]As outlined in the DBpedia ontology page `http://web.archive.org/web/20210416134559/http://wikidata.dbpedia.org/services-resources/ontology`

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

inns, or any other establishments offering temporary accommodation to the public[33];

2. it differentiates between lodging facilities and specific accommodations provided within these establishments;

3. it incorporates an expanded hierarchy[34] of types of lodging facilities (e.g., hotels, houses, resorts) and accommodations (e.g., hotel room, entire apartment);

4. it features an extended hierarchy of amenities (for example, air conditioning, tv, parking) offered by lodging facilities;

5. it includes an extended hierarchy of geographic features pertinent to tourism (based on Schema.org) and enhanced with GeoNames feature taxonomy (utilizing the GeoNames mapping[35] data-set);

6. it employs Schema.org for modeling tourist reviews;

7. it utilizes Schema.org to represent Tourist Destinations and Tourist Locations;

8. it can be conveniently extended to encapsulate other types of entities significant to tourism in the future (such as events or restaurants).

As can be seen in Figure 3.2, the TAO ontology schema is shown in a way that allows for repurposed classes from the previously mentioned ontologies to be easily identified. For future reference, we will be using the `tao:` prefix when referring to TAO. Key classes in the ontology include `tao:LodgingFacility` and `tao:Accommodation`, which are used to represent lodging facilities and their accommodations, respectively. The `tao:LodgingFacility` class is associated with the lodging business concept utilised in Schema.org, but only pertains to the physical location which houses the facilities' accommodations (for instance, a hotel is viewed as the building containing rooms). This ensures a clear distinction between the physical location and the business organisation that owns or manages the lodging facility, preventing any inconsistencies arising from GoodRelations disjunction between `schema:Place` and `schema:Organization` classes, as mentioned in Section 3.4.3. Descriptions of facility locations are provided based on their latitude and longitude literal properties, as well as by utilising the `schema:PostalAddress` class, which allows for an practical specification of the address. To round off the description of the facility, we use literal properties to indicate its name (`schema:name`) and a relevant online resource (`schema:mainEntityOfPage`). The object property

[33]Definition sourced from Law Insider, see `https://www.lawinsider.com/dictionary/lodging-facilities`

[34]We use the term "hierarchy" to refer to a subsumption hierarchy of concepts, similar to the one utilized by DBpedia, which involves a hierarchy of classes connected with rdfs:subClassOf property.

[35]`https://www.geonames.org/ontology/mappings_v3.01.rdf`

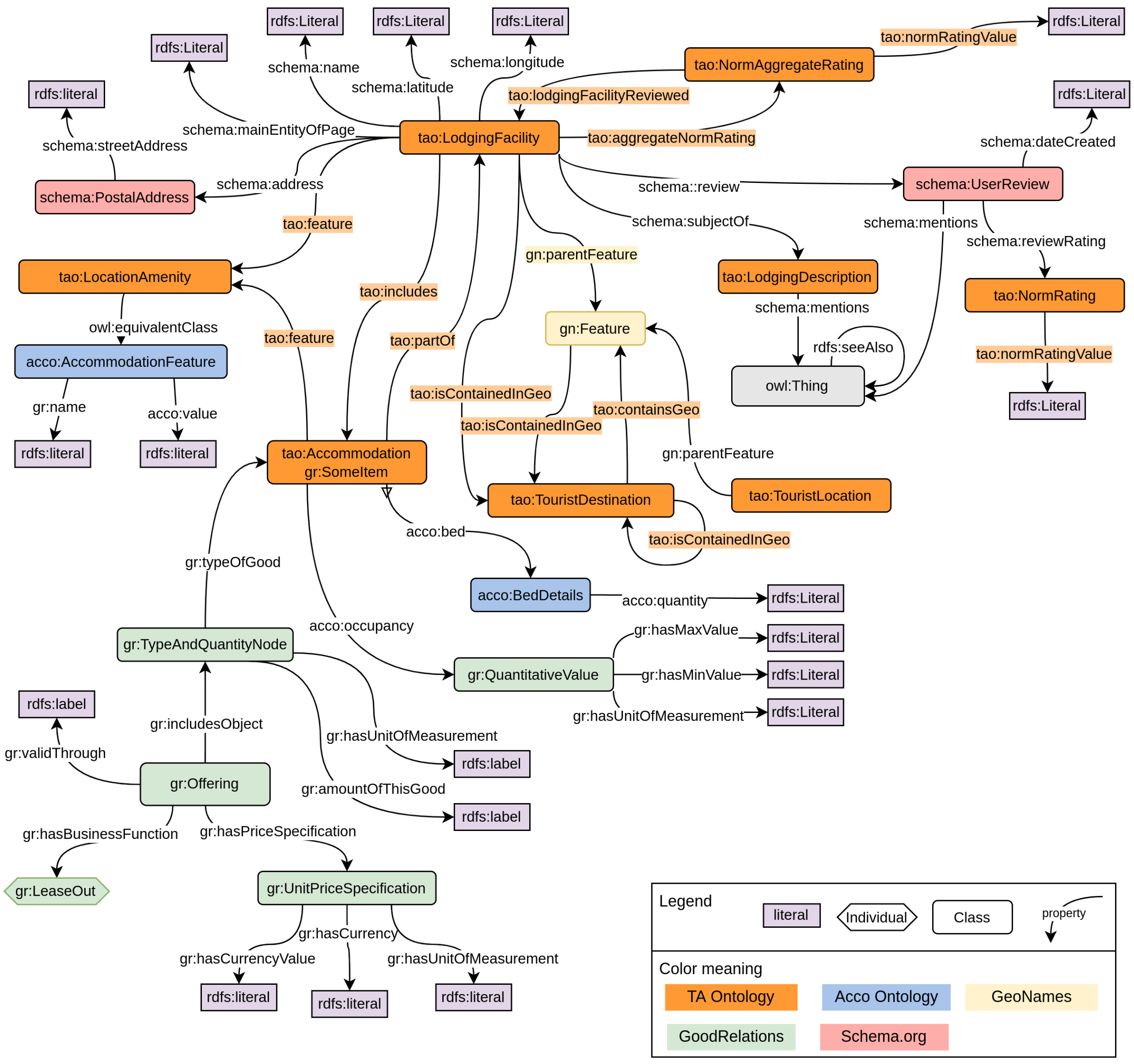


Figure 3.2: This is a depiction of the TAO ontology schema. It is important to note that in this schema, each arrow is symbolic of a semantic relationship, initiating from its domain and concluding in its range.

tao:aggregateRating can be used to link a lodging facility to an overall rating, represented by a tao:NormAggregateRating node and annotated with the data property tao:normRatingValue to denote a float value between 0 and 1. A lodging facility can also be linked, via the schema:subjectOf property, to a textual description represented by the tao:LodgingDescription class. Finally, lodging facilities can be associated, using the schema:review property, to one or more user reviews, represented using the schema:UserReview class. Each review is defined by its creation date and linked, using the schema:reviewRating property, to a rating (vote) represented by the tao:NormRating class, which can be used to specify the normalised rating in a specific review. The facility description and the reviews can mention any type of entity, including those defined in other knowledge graphs (DBpedia and GeoNames) using the schema:mentions property. This data is typically derived from the text of descriptions and reviews using various entity linking techniques, such as DBpedia Spotlight, Mordecai, OpenTapioca, or Falcon. Entity linking refers to the process of associating text fragments with their corresponding entities in a knowledge graph.

The tao:Accommodation class, similar to schema:Accommodation, represents the actual units of the lodging facility that are available for rent. It is formally separate from the physical building where the accommodations are located, which is represented by the tao:LodgingFacility class. TAO uses the tao:includes object property to establish the relationship between a lodging facility and its accommodations. For the TAO ontology to remain somewhat compatible 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. As a result, if a node in the KG is a member of tao:Accommodation it is also a member of acco:Accommodation, and all properties defined in the Accomodation ontology for accommodations still hold true. However, not all nodes that are members of acco:Accommodation are also members of tao:Accommodation.

In accordance with 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 as type gr:SomeItem[36];

- the offering is represented by a gr:Offering node, which has a validity period defined by the gr:validThrough data property and is characterised by a specific business function using gr:hasBusinessFunction to denote that it is a gr:LeaseOut[37];

[36]It is also of type tao:Accommodation

[37]Defined in the GoodRelations ontology as an individual of type gr:BusinessFunction.

- 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 (via the `gr:typeOfGood` relation), the quantity of the good included (using `gr:amountOfThisGood` data property) and the unit of measure for the quantity 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 is included in the price (using `gr:hasUnitOfMeasurement`) i.e., a DAY in the accommodation.

The `acco:occupancy` property, along with a `gr:QuantitativeValue` node, can be used to define the occupancy for a particular accommodation. The "C62" literal, which represents "one piece" of something (in this case a person), is specified through the `gr:hasUnitOfMeasurement`[38]. The minimum and maximum number of people allowed are defined using the `gr:hasMinValue` and `gr:hasMaxvalue` respectively. The `acco:BedDetails` class and the `acco:quantity` property are used to specify the number of beds. This method is similar to the one employed by the Accommodation Ontology. TAO utilises the `tao:LocationAmenity` class to represent an amenity provided by an accommodation or lodging facility. This class is equivalent to `acco:AccommodationFeature` for compatibility with the Accommodation Ontology. Additionally, the `tao:feature` property is used to associate an accommodation or lodging facility with one or more amenities.

A tourist location, such as the London Tower or the Poetto beach in Cagliari, is represented using the `tao:TouristLocation` class. This class is a subclass of both `schema:Place` and `gn:Feature`. Similarly, a tourist destination, such as Sardinia, is modelled using the `tao:TouristDestination` class. This class is equivalent to `schema:TouristDestination` and a subclass of `gn:Feature`. The `tao:isContainedInGeo` property is used to include tourist locations and lodging facilities within a tourist destination. For example, if the City of London is included in a tourist destination, all `tao:LodgingFacility` individuals in the City of London (according to the `gn:parentFeature` property) are also considered part of that destination. This is due to an axiom in the TAO ontology that defines a chain of properties. According to this chain, if `X gn:parentFeature Y` and `Y tao:isContainedInGeo Z`, then `X tao:isContainedInGeo Z`. This can be expressed in functional-style syntax as:

```
   SubObjectPropertyOf( ObjectPropertyChain( gn:parentFeature
tao:isContainedInGeo ) tao:isContainedInGeo ).
```

[38]`http://www.heppnetz.de/ontologies/goodrelations/v1#UnitPriceSpecification`

TAO also includes several subsumption hierarchies that describe the relationships between relevant classes. These include:

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

Appendix C provides a comprehensive explanation of these four hierarchies.

TAO enrichment through code

We constructed the TAO ontology by employing a programmatic method as opposed to manual editing. In particular, we designed a building process using Python and the owlready2 [126] library. While other methods like templates (OPPL [127], OTTR [128]) or alternative languages (such as Tawny-OWL [129] which utilizes Clojure) are available, we opted for Python, a full programming language that is also highly suitable for data manipulation and transformation. This decision also enabled us to use widespread software engineering tools and practices, automating parts of the ontology building process (e.g., generating axioms), version the code rather than just the final ontology, decrease human mistakes, and conveniently create inline documentation about the ontology creation process. We also offer an open-source version of the Python code that builds the TAO ontology as a Jupyter Notebook[39].

TAO ontology must be capable of modeling data gathered from common sources in the tourism industry, such as Booking.com and Airbnb. These platforms provide (semi)structured data as key/value properties and unstructured data as text relating to lodging facilities, accommodations, amenities, and user reviews. As such, we designed a human-in-the-loop strategy, depicted in Figure 3.3, to continuously enhance TAO by constantly adding new types of `tao:LodgingFacility`, `tao:Accommodation` and `tao:LocationAmenity` or new labels for existing types derived from the source data. This method allows us to maintain the ontology current and properly aligned with the actual data.

We kick off with the basic version of the ontology (orange bullet 1 in the figure), prepare external imports, and establish classes, properties, and axioms (bullet 2). To further embellish TAO, our ontology engineers, in cooperation with domain experts, assess various analytics about the most frequent terms associated with facilities, accommodations, and amenities. They then use these to form new relevant classes in the ontology (bullet 5) or apply additional labels to an existing class (bullet 6). We can give two examples: the mini-golf amenity class was extracted from Booking.com amenities, and "holiday house"

[39]See `https://github.com/linkalab/tkg/tree/main/tao_modelling`

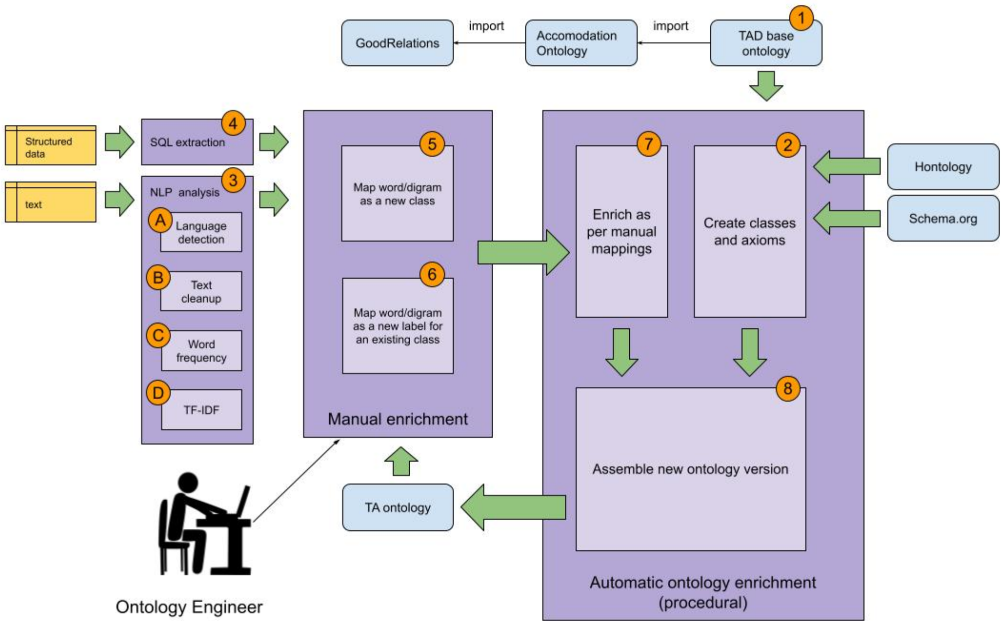


Figure 3.3: Ontology enrichment workflow.

was added to holiday home lodging facility class as an alternative label extracted from Airbnb texts. We implemented two automatic pipelines (3 and 4) to produce the necessary analytic from the underlying data. The first one extracts and ranks frequent uni-grams and bi-grams from lodging facilities descriptions and reviews. To accomplish this, we used the Spacy Python library[40] to perform the following tasks: 1) filter English text only using language detection algorithms (bullet A), 2) remove special characters from text (bullet B), 3) produce a frequency analysis of words used in texts (bullet C), and 4) produce a TF-IDF analysis of all texts (bullet D). The second pipeline handles structured data, extracting a list of all distinct values for categorical fields related to lodging facilities and accommodation types, or accommodation features.

In the end, the ontology engineers generate a mapping file that is used (bullet 7) to form new classes, establish subclass relations (with `rdfs:subClassOf` property) or enrich existing classes with new alternate labels (with the `skos:altLabel` property). We also document the source of these changes directly in the ontology by using properties `dc:source` for classes and `rdfs:comment` for labels. The final step (bullet 8) generates a new version of the TAO ontology.

In Appendix F, we present some code snippets to illustrate the iterative extension of the TAO ontology.

[40]`https://spacy.io/`

3.5 Knowledge Graph generation phases

This section will outline the final three steps involved in constructing the Knowledge Graph, as shown in Figure 3.1.

3.5.1 Data Transformation

The fourth phase of our process in creating our Tourism Knowledge Graph involves transforming the data. This specific phase entails converting the information gathered from the different data sources into a series of tables. These tables are then used in the subsequent phase (outlined in Section 3.5.2) to create the actual knowledge graph triples. This section is dedicated to explaining the process of data transformation and the technologies used in its execution. Depending on the structure of the source data and the required outcome, we can implement a variety of transformation steps arranged as data pipelines. A data pipeline refers to a sequence of computational steps organized as a direct acyclic graph where the result of one step becomes the input of one or more subsequent steps.

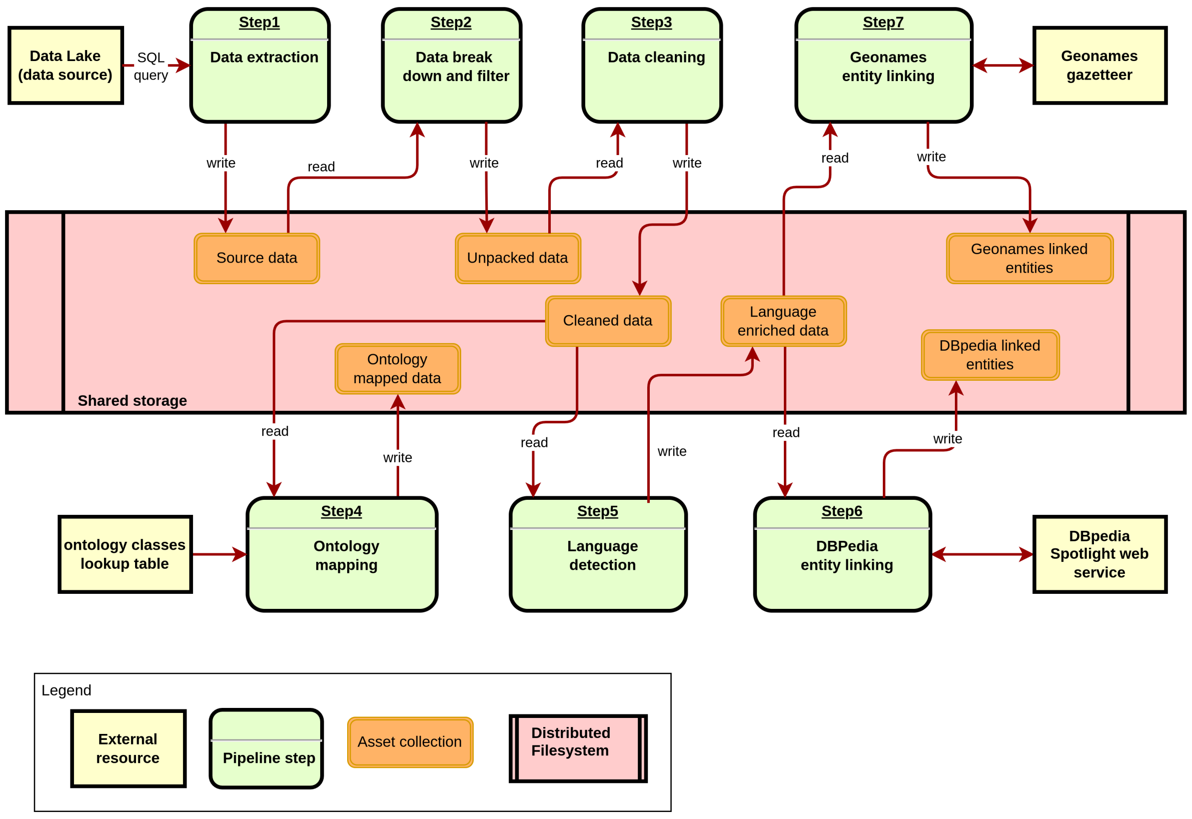


Figure 3.4: High level data transformation workflow diagram.

Figure 3.4 provides a visual representation of the full data transformation workflow.

Each step can materialize its output (referred to as an *asset* from this point forward), either saving it as a file or storing it within a database application. From the diagram, four types of components can be observed:

1. external resources utilized during the pipeline execution (yellow boxes) which represent
   (a) data lake tables,
   (b) ontology classes lookup tables stored as files that map text strings to TAO ontology classes,
   (c) the DBpedia Spotlight public web service,
   (d) GeoNames gazetteer presented as an Elasticsearch endpoint;
2. the pipeline execution steps (green boxes);
3. groups of data assets (files) produced by the execution steps (orange boxes);
4. a distributed file system that stores all the data assets created and utilized by one or more processing steps (pink box).

Broadly speaking, the workflow comprises seven steps. The first three steps are executed on both semi-structured (key/values) and unstructured (text) data:

1. **Data extraction**: connects to sources and extracts all contained data producing the *Source data assets collection*;
2. **Data break down and filter**: reshapes the data structure also filtering out unnecessary data; works together with the Data cleaning step producing the *Unpacked data assets collection*;
3. **Data cleaning**: corrects/removes corrupt, duplicated or inaccurate data from *Unpacked data assets collection* producing the *Cleaned data assets collection*;

The pipeline handles in a different way structured and unstructured data contained in the *Cleaned data assets collection.* In case of structured data, the final step is:

4. **Ontology mapping**: utilizes heuristic rules to determine which ontology class should be used to model each entity contained in the data; its output is the *Ontology mapped data assets collection*;

In the case of unstructured data, our goal is to enhance TKG with links to semantic entities from DBpedia and GeoNames cited in lodging descriptions and user reviews. This enrichment will enable TKG to connect to external knowledge graphs showcasing what tourists and business owners deem important and noteworthy. To accomplish this enrichment, we perform entity linking, in three more steps:

5. **Language detection**: recognizes the language used in texts to process only English text; produces the *Language enriched data assets collection*;

6. **DBpedia entity linking**: produces the *DBpedia linked entities data assets collection* by recognizing and linking DBpedia entities cited in descriptions and reviews;

7. **GeoNames entity linking**: produces the *GeoNames linked entities data assets collection* by recognizing and linking GeoNames entities cited in descriptions and reviews.

In Appendix D, we delve into each processing step and the technology architecture used.

3.5.2 Forming Triples

The fifth phase in the creation of the Travel Information Graph, as shown in Fig. 3.1, involves generating RDF triples. To achieve this, we exploited the RDF Mapping Language (RML) [98] to establish data pipelines that would generate RDF triples[41] from text documents and subsequently save them in a serialized format. RML is a declarative language used to specify the process of creating Linked Data from corresponding data sources using annotations provided through vocabulary terms. RML can also utilize files as data sources, making it particularly useful for our scenario. The following elements[42] are required for an RML transformation:

1. an RML processor that carries out the transformation;

2. a data source that serves as the input to the RML mapping;

3. an RML mapping that outlines the rules for converting any input (structured) data to RDF.

These rules outline the process of converting an input record (CSV row, XML element, or JSON object) into one or more RDF triples. They are not tied to the execution process of the conversion, thereby separating the rules from the implementation.

In our execution, we made use of RMLMapper [99], an open-source RML processor written in Java[43]. Different mappings were designed to deal with the different sources, such as Booking.com and Airbnb. The RML processor's output are files containing the serialized RDF triples in n-quads[44]. To streamline the development, debugging, and maintenance of RML triple maps, we employed YARRRML [130], a text-based, human-readable representation for declarative generation rules[45]. In the paragraphs that follow, we will explore an example of how a Lodging Facility and all related entities can be represented in TKG using a set of triples generated through the above process. We will use graphical representation to better understand the structure of the knowledge graph.

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

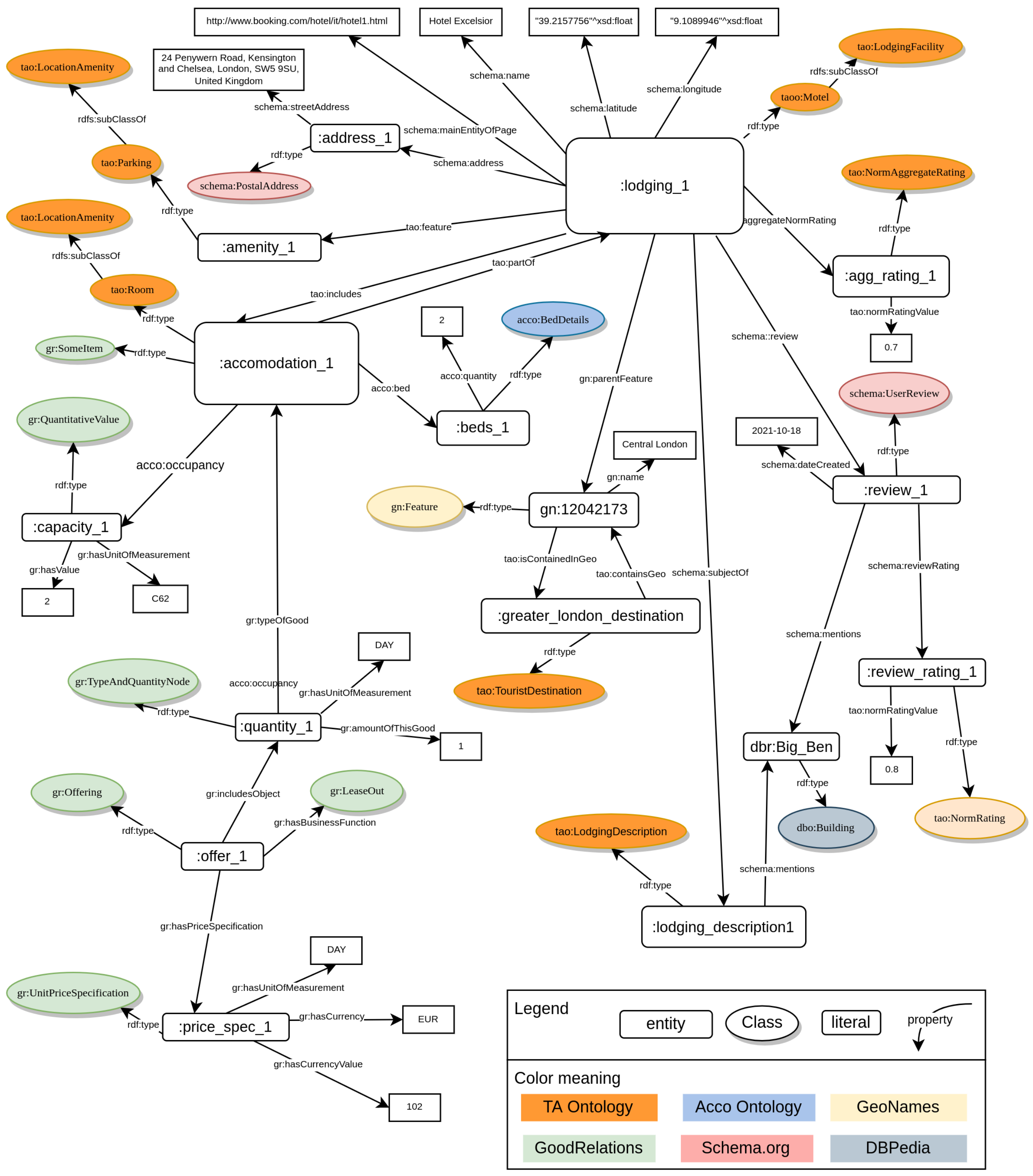


Figure 3.5: A high-level example of the main entities used in TKG.

High-level triples structure for Tourism Knowledge Graph

The process of creating triples to describe accommodation offers adheres to the rules of the Accommodation Ontology and complies with the best practices of GoodRelations and Schema.org. Figure 3.5 displays a high-level example of the TKG structure. Here we see a lodging facility (`:lodging_1`) which is the subject of a descriptive text (`:lodging_description_1`), has one review (`:review_1`), and includes one accommodation (`:accommodation_1`). A description is a type of creative work (modeled using the `tao:LodgingDescription` class) that can refer to one or more real entities such as places or food. In this example, the description mentions the Big Ben tower (via the `schema:mentions` property). Reviews are also considered creative works in Schema.org[46] and are thus related to entities cited by users in their texts using the `schema:mentions` property. This is how DBpedia and GeoNames entities, produced by the entity linking processes, are integrated in TKG. There is an offer (`:offer_1`) to lease an accommodation (`:accommodation_1`) through the (`:quantity_1`) node. The properties of this node define what is being offered using the `gr:hasUnitOfMeasurement` property (e.g., DAY) and in what quantity using the `gr:amountOfThisGood` property (e.g., 2). The accommodation is contained within the lodging facility; `:offer_1` using property `tao:includes`.

In Appendix E, we provide a more detailed explanation of the structure of triples representing lodging facilities, accommodations, offers, and user reviews in the Tourism Knowledge Graph.

3.5.3 Knowledge Graph publishing and validation

This segment will outline the process of publishing TKG with a triple store, clarify how we distinguish the various resources in the knowledge graph, and finally detail how we set up the provenance. We dedicated Section 3.6 to present the validation of the TKG with respect to the identified use cases together with a more general evaluation of TAO ontology in comparison with other solutions.

We utilized Ontotext GraphDB for the publishing of the knowledge graph. The knowledge graph is essentially a set of numerous RDF graphs. Every RDF graph has a corresponding IRI which establishes its graph name. We developed two types of named graphs for both Booking.com and Airbnb:

1. hospitality named graph, which encompasses all the triples produced using data resources created at the conclusion of the ontology mapping step[47], processing semi-structured data extracted from a specific source (such as Airbnb) for a particular tourist destination (in our case, London or Sardinia);

2. linked entities named graph, which includes all the triples constructed using data resources created at the end of the DBpedia or Geonames entity linking steps[48], processing texts extracted from a specific source (such as Airbnb) for a particular tourist destination (in our case, London or Sardinia).

[46]See `https://schema.org/UserReview`
[47]as detailed in Appendix D.
[48]as outlined in Appendix D.

A named graph has a custom IRI with the following structure:
`base_url/tourist_destination/source/enrichment`
More in detail:

1. base_url: `http://tourism.kg.linkalab-cloud.com/ng/`[49]
2. tourist_destination: is used to distinguish a tourist destination by name (such as London or Sardinia)
3. source:
   (a) bkg: is used to distinguish the source Booking.com ;
   (b) air: is used to distinguish the source Airbnb;
4. enrichment:
   (a) internal: is used for all the RDF resources that are created without entity linking in the transformation phase;
   (b) dbpedia_el: on resources that are enriched with Entity Linking using DBpedia;
   (c) geonames_el: on resources that are enriched with Entity Linking using GeoNames.

For instance, the named graph which is a collection of triples about Sardinia's hospitality, produced from Airbnb (semi-)structured data (with no entity linking) would have this IRI: `http://tourism.kg.linkalab-cloud.com/ng/sardinia/air/internal`.

The implementation of named graphs as described above aids in distinguishing resources related to a particular tourist destination since we can employ the named graphs in SPARQL queries and use Triple Pattern Fragments[50] (TPF) [131, 132] to identify data subsets through Implicit Graphs. This distinction is also beneficial for expressing provenance metadata at the named graph level as outlined in Section 3.5.3.

In terms of identifying a resource in the knowledge graph, we utilize IRIs that explicitly include the external source (such as Booking, Airbnb), and the resource type. The IRI of a resource is structured in the following way: `base_url/resource_type/source/unique_id`

More in detail:

1. base_url: `http://tourism.kg.linkalab-cloud.com/`
2. resource_type:
   (a) lf: is used to distinguish Lodging Facility entities;
   (b) ac: is used to distinguish Accommodation entities;
   (c) of: is used to distinguish Offering entities;
   (d) rv: is used to distinguish User Reviews entities.

[49]ng stands for named graph.
[50]`http://linkeddatafragments.org/`

3. source:
    (a) bkg: the resource comes from Booking.com;
    (b) air: the resource comes from Airbnb.
4. unique_id: is an identifier created during the data transformation phase which is unique to the data source.

For example, the following IRI identifies a lodging facility that comes from Airbnb: `http://tourism.kg.linkalab-cloud.com/lf/air/30840569`.

The Tourism Analytics ontology is serialized as an RDF/XML file published at: `http://purl.org/tao/ns`[51]. Each specific class or property in the ontology is accessible using the hash IRI standard[52] (for example, `http://purl.org/tao/ns#LodgingFacility` is the IRI for the LodgingFacility class).

Origin and metadata of the dataset

We also loaded the metadata triples that detail the other named graphs and their origin into a specific named graph: `http://tourism.kg.linkalab-cloud.com/ng/meta/prov`. A named graph can be referred to using Quad Pattern Fragments[53] with a IRI having the following structure: `base_url?graph=graph_name` where the elements are:

1. base_url: `http://tourism.ldf.linkalab-cloud.com/graph`
2. graph_name: is the IRI that identifies the named graph

For instance, the named graph containing the triples about the London hospitality industry derived from Booking.com's (semi-)structured data (without entity linking) would be referred to as:

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

We utilized the W3C PROV[54] provenance model to express the origin information. This enables us to trace the lineage of data assets created during the data transformation and triple creation phases, following an approach similar to the one introduced in [99] and embedded in the RMLMapper tool. In particular we employed the Implicit Graphs approach to express metadata at the Named Graph detail level of granularity, in order to produce a small number of additional provenance triples. In this setting, the time taken to generate metadata is insignificant compared to the overall triple generation time, similar to what can be experienced when using the RMLMapper metadata generation feature with a comparable configuration.

In PROV, we have three primary classes:

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

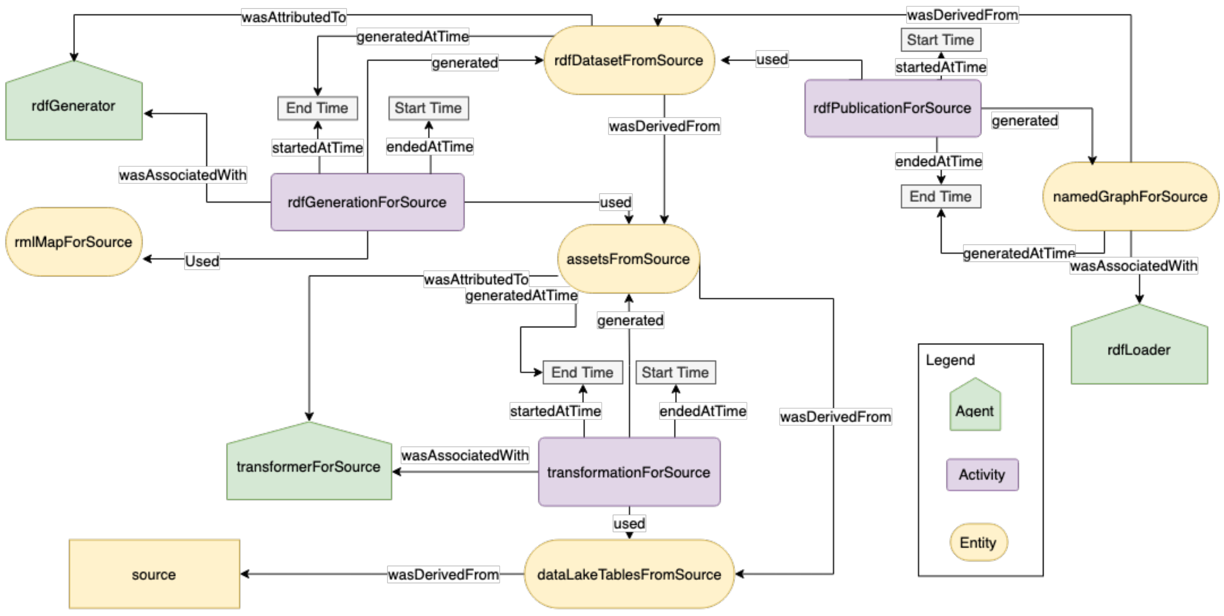


Figure 3.6: A high-level provenance metadata schema for a named graph in the Tourism Knowledge Graph.

- `prov:Entity` - is used for entities which can be real or fictional, physical, digital, conceptual, or of other kind;
- `prov:Activity` - an event that happens over a period of time and interacts with or affects entities; it might consume, process, transform, modify, relocate, utilize, or generate entities;
- `prov:Agent` - an entity bearing some kind of responsibility for an activity occurring, for an entity's existence, or for another agent's activity.

Figure 3.6 presents a high-level origin schema that describes how origin metadata for a specific named graph is structured. Specifically, the following PROV entities can be identified:

1. `source` - signifies the web source of our data (e.g., Booking.com);
2. `dataLakeTablesFromSource` - signifies the tables in the data lake containing the data extracted from the source;
3. `assetsFromSource` - signifies all the assets created during the transformation phase that are utilized to produce the RDF triples for a specific named graph;

4. `rmlMapForSource` - signifies the RML map document employed to produce the RDF triples for a specific named graph;
5. `rdfDatasetFromSource` - signifies the RDF graph (serialized as one or more files) that is produced from the source using specific `assetsFromSource` and `rmlMapForSource` entities;
6. `namedGraphForSource` - signifies the published named graph.

In addition, in the same schema, the following PROV Activities involved in the production of a specific named graph can be identified:

1. `transformationForSource` - executed to prepare/enrich the data for the creation of triples;
2. `rdfGenerationForSource` - executed to create the triples;
3. `rdfPublicationForSource` - executed to write the triples into the triple store as named graphs.

Lastly, the following PROV Agents can be identified in the schema:

1. `transformerForSource` - signifies the entire transformation pipeline described in Section 3.5.1;
2. `rdfGenerator` - signifies the RML processor software (RMLMapper in our case);
3. `rdfLoader` - signifies the agent responsible for loading the RDF graph into the triple store.

The suggested PROV schema can be readily adapted to specify a particular named graph's origin information and can track: (i) time information about the creation and update of all triples in the named graph, (ii) the assets used to create the triples, (iii) the RML mapping document applied to generate them. The same metadata can be produced for all the assets the transformation pipeline produces. Moreover, the agent entities can trace which software version was used to produce a specific named graph. It is important to note that we decided to use the described origin schema and a custom metadata generator, instead of using RMLMapper software capabilities, because we wanted to keep track of all the pipelines (data transformation, RDF generation, and RDF publication) using a unified approach by leveraging the selected orchestration service (Dagster) as described in Appendix D.8.

3.6 Validation and assessment of TKG and TAO

TKG and TAO were assessed[55] based on functional, logical, and structural metrics as recommended by prior research [133, 134]. The *functional dimension* pertains to the ability

[55]It's important to mention that TAO was also validated using OOPS! (`https://oops.linkeddata.es/`) to detect and rectify common errors. We manually reviewed the output of the tool and, after disregarding issues related to other ontologies or incorrect results, we pinpointed and rectified 47 missing annotations, 3 missing domain and range specifications in object properties, 1 incorrect equivalent class definition, and 3 inverse relationships that weren't explicitly declared.

to meet the requirements and provide a useful depiction of the tourism sector, whereas the *logical dimension* relates to the capacity to be accurately processed by a reasoner and generate valid new knowledge. Together these dimensions allows for a validation of the knowledge graph and the supporting ontology as prescribed by the methodology proposed in this work.

Both functional and logical dimensions were assessed by devising and executing a series of tests. We developed the test cases as RDF files modeled with the TestCase OWL metamodel (prefix `test:`), as per Blomqvist et al. [135]. Each test case outlines its inputs, execution conditions, the actual testing process, and the expected outcomes. All resultant RDF files are accessible at `https://github.com/linkalab/tkg/tree/main/validation`. Finally, the review of the *structural dimension* is aimed at examining the topological characteristics of TKG and TAO, which is also compared with other ontologies (e.g., Hontology and Acco). These results provide valuable feedback on design decisions and can be employed to iteratively fine-tune the knowledge graph. Detailed information are provided in the following three subsections.

3.6.1 Functional assessment

To ensure the functional requirements are met, we utilized the **CQ (Competency Question)**[56] verification method proposed by Carriero et al. [134]. In practice, this method aims to check if the competency questions can be translated into SPARQL queries and executed on the KG. For this, we devised 12 test cases by converting the competency questions, detailed in Appendix B.2, into SPARQL queries. The input data were extracted from the knowledge graph to examine each specific functionality. This process guided the creation and refinement of TAO, discovering missing classes or properties and integrating them into the ontology. It was also used to ensure that TKG can respond effectively to all competency questions.

The execution of each test case involves running the respective query against the TKG SPARQL end point[57]. Queries were manually executed and the outcomes were compared with the expected results. Some CQs necessitated the execution of federated queries to retrieve triples from DBpedia and GeoNames. For this, we utilized the `SERVICE` keyword to access Ontotext FactForge SPARQL endpoint[58].

All 12 competency question tests were successful. The following example (Listing 3.6.1) demonstrates a federated SPARQL query designed to answer *"What are the apartments with wi-fi near at least 2 parks?"*[59].

```
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#>
```

[56] A Competency Question is a query articulated in natural language as detailed in Section 3.4.3.

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

[58] Refer to `http://factforge.net/`

[59] Here, a park is deemed close to the apartment if it's within a distance of 1 km.

```
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 produced the following results.

Lodge	**Apartment**	**Near_parks**
`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 can be found at `https://github.com/linkalab/tkg/tree/main/validation/competency_questions`[60]

3.6.2 Logical Assessment

In order to evaluate the logical dimension, we initially checked for any inconsistencies executing a reasoner on TAO. Following this, we carried out a full assessment of the TKG following two strategies recommended by Carriero et al. [134]:

[60]We recommend using Protégé to open the competency questions test cases files.

1. **Inference Verification**, which inspects whether the inference over the KG yields the anticipated results (for instance, if an accommodation of type `tao:HotelRoom` is contained in a `tao:LodgingFacility`, it can be inferred that the latter is a Hotel);
2. **Error Provocation**, which seeks to induce an inconsistency error by inputting data that violates ontology axioms (for example, an instance of a lodging facility cannot be simultaneously a `tao:Hotel` and a `tao:BedAndBreakfast`).

We therefore develop the appropriate test cases to determine what inferences can be made and what kinds of errors the reasoner might produce. In this scenario, we cannot depend on CQs and must instead inspect the ontology structure and contemplate how classes and properties are defined by axioms. In the subsequent subsection, we will delve deeper into how we performed these two tests.

Inference Verification

To evaluate this dimension, we created 15 test cases as OWL files, utilizing the TestCase OWL meta-model. These files are assigned a unique IRI and only contain the ABox (Assertional Box), depending[61] on the TBox (Terminological Box)[62] of the TAO ontology and the TestCase metamodel[63]. The ABox contains a set of individuals necessary to perform the test and achieve the anticipated results. All inference verification test cases and the associated data sets can be found at `https://github.com/linkalab/tkg/tree/main/validation/inference_verification`.

These tests are beneficial to ascertain whether the ontology can be effectively used to extend the knowledge graph with reasoning, for instance, using inverse properties definitions we can materialize backlinks[64], or using a chain of object properties we can infer new relationships[65], or we can infer the type of an entity from its properties[66]. As a practical scenario, consider a `LodgingFacility` individual (named `Hotel Splendor`) which is included inside `Greater London`, defined in GeoNames[67] as a second-level administrative division, through the ObjectProperty `gn:parentADM2`. Suppose that there exists a `TouristDestination` individual called `GreatLondonDestination` which includes (via the `tao:containsGeo` property) Greater London. In this case, the reasoner should infer that `Hotel Splendor` is also a part of `GreatLondonDestination`.

We conducted the final evaluation by loading the test files in Protégé software[68] and executing the Pellet reasoner[69]. All 15 test cases produced the expected results.

[61]Using `owl:imports`.

[62]ABox and TBox where introduced in Section 2.3.6.

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

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

[65]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.

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

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

[68]See `https://protege.stanford.edu/`

[69]See `https://github.com/stardog-union/pellet/tree/master/protege/plugin` for

Error Provocation

This test seeks to comprehend how the knowledge graph (TKG) responds to the introduction of inconsistent data. For instance, since an entity cannot be simultaneously a `tao:Hotel` and a `tao:BedAndBreakfast`, if we introduce an individual which is defined as belonging to both classes we should find a validation error. The test is successful if the reasoner detects an inconsistency because the appropriate disjointedness axiom is violated. We followed the same approach used in the verification of inference tests described above. Additionally, for some tests, we also created a SHACL file defining additional constraints[70].

We created 12 test cases for error provocation, testing whether incorrect patterns in the knowledge graph are detected. For example, it is not correct to include hotel rooms in a lodging facility that is not a hotel, or to include the same accommodation to multiple disjoint lodging facilities, or if we do not connect an amenity to any accommodation or lodging facility[71]. We loaded each test file within Protégé, and then we ran both the reasoner and the SHACL rules engine[72]. A test is successful if the introduced inconsistencies are identified by the reasoner and/or the SHACL validator.

We employed this same technique of error provocation to test the correct creation of triples during the triple creation process (see section 3.5.2) and to refine axioms and constraints in TAO. All error provocation test cases and the associated data sets can be found at `https://github.com/linkalab/tkg/tree/main/validation/error_provocation`.

3.6.3 Structural aspect

The structural aspect of TAO and TKG was evaluated by calculating various metrics that are used to assess ontologies and KGs, as defined and applied in previous studies [133, 134]. We adopted a similar methodology to Carriero et al. [134], which took into account both fundamental and topological metrics. Basic metrics are employed to evaluate the subsequent quantitative elements:

- *number of axioms* - the complete count of axioms set for classes, properties, datatype definitions, assertions, and annotations;
- *number of logical axioms* - the count of axioms that influence the logical interpretation of an ontology;
- *number of classes* - the complete count of classes set in the ontology;
- *number of object properties* - the complete count of object properties set in the ontology;

Protégé plug-in named Pellet reasoner.

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

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

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

- *number of datatype properties* - the complete count of datatype properties set in the ontology;
- *number of annotation assertions* - the complete count of annotations in the ontology;
- *DL expressivity* - the description logic expressivity of the ontology.

Conversely, topological metrics are beneficial for comprehending ontology richness, width/depth, inheritance structure, cohesion, and multi-hierarchical degree.

Specifically, we employed the subsequent metrics:

- *Inheritance Richness (IR)* - calculates the average count of sub-classes per class[73]. Low figures suggest a vertical (deep) ontology whereas high figures suggest a horizontal (shallow) ontology.
- *Relationship Richness* - calculates the ratio of the count of non-inheritance relationships divided by the count of relationships of all types[74]. Figures are standardised to one, where 0 suggests that only inheritance relations are present in the ontology and 1 that no inheritance relations exist.
- *Axiom Class Ratio* - calculates the ratio of the count of axioms divided by the count of classes[75]. An ontology with minimal axiomatisation has a low value of this metric (near zero); higher figures suggest a better axiomatisation, but extremely high figures can indicate an excessive axiomatisation.
- *Class/property ratio* - calculates the ratio of the count of classes divided by the count of relations[76]. Low figures (i.e., ~ 0) are observed in ontologies with numerous properties connecting a few concepts. In contrast, high figures suggest that the ontology has numerous classes connected by few properties.
- *NoR* - count of root classes (a class which is not a subclass of other classes)[77]. The interpretation of NoR depends on the total count of classes. We reveal (i) the ordinal figures of NoR and (ii) the ratios between NoR and the count of classes in parenthesis.
- *NoL* - count of leaf classes (all classes that have no sub-classes)[78]. The interpretation of NoL depends on the total count of classes. We reveal (i) the ordinal figures of NoL and (ii) the ratios between NoL and the count of classes in parenthesis.

[73]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Inheritance_Richness`

[74]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Relationship_Richness`

[75]`https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Axiom_Class_Ratio`

[76]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Schema_Metrics#Class_Relation_Ratio`

[77]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Number_of_root_classes_.28NoR.29`

[78]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Number_of_leaf_classes_.28NoL.29`

- *NoC* - count of external classes[79] defined by [136]. A low figure of NoC can suggest that the ontology is semantically independent; a high figure can suggest that the ontology depends on concepts defined in other ontologies. The interpretation of NoC also depends on the count of classes in ontology. We reveal (i) the ordinal figures of NoC and (ii) the ratios between NoC and the count of classes in parenthesis.
- *ADIT-LN* (Average depth of inheritance tree of leaf nodes) - is the average depth of the graph where classes are nodes and `subClassOf` properties are arcs [80].
- *Max breadth* - the maximum breadth of the inheritance tree[81]. The figure of *Max breadth* should be considered in relation to the count of classes in ontology.
- *Average breadth* - the average breadth of the inheritance tree[82].
- *Max depth* - the maximum depth obtained by traversing the inheritance tree.[83] The figure of *Max depth* should be considered in relation to the count 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 inheritance tree[84]. A figure of 0 indicates no tangledness; a figure of 1 indicates that each class has multiple super-classes.

Tables 3.1, 3.2, and 3.3 present the base and topological metrics for TAO, Hontology, and the Accommodation Ontology (Acco). It's worth pointing out that our analysis of TAO only took into account the classes and properties unique to this ontology and excluded those imported from other ontologies (i.e., the Accommodation Ontology, GoodRelations). This decision was made to ensure a fair comparison with the Accommodation Ontology, which we import. We used the OntoMetrics[85] web application to produce all metrics.

Table 3.1 indicates that TAO is considerably larger than Hontology and the Accommodation Ontology, in terms of the quantity of classes, axioms, logical axioms[86],

[79]A class is considered external when it is defined in a different ontology. This metric has been calculated using Protégé.

[80]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Knowledgebase_Metrics#Average_depth_of_inheritance_tree_of_leaf_nodes_.28ADIT-LN.29`

[81]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Maximal_breadth`

[82]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Average_breadth`

[83]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Maximal_depth`

[84]See `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Graph_Metrics#Tangledness`

[85]Refer to `https://ontometrics.informatik.uni-rostock.de/ontologymetrics/index.jsp`

[86]Logical axioms influence the logical interpretation of an ontology. Refer to `https://ontometrics.informatik.uni-rostock.de/wiki/index.php/Base_Metrics#Logical_Axiom`. Conversely, non-logical axioms, like entity declarations or annotations, do not impact the

Table 3.1: 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 3.2: 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]	*not provided*[7]
Tourist location types	146	22[4]	*not provided*[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 3.3: 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

and annotation assertions. Most of the additional classes describe various kinds of lodging facilities (35 classes), accommodations (17 classes), amenities (343 classes), and tourist sites (146 classes). Table 3.2 provides a comparison of these classes. TAO has a greater variety of lodging facilities, amenities, and tourist locations compared to Hontology and Acco. While Hontology appears to have more accommodation types, this could be attributed 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 context, we chose not to include specific sub-classes, instead using amenities (e.g., `acco:Terrace`) and bed detail specifications (using `acco:BedDetails`) for a better characterization of accommodations. When it comes to properties, TAO only introduces a handful of new ones, as it primarily reuses those from Acco (4), GoodRelations (15), Schema.org (11), and GeoNames (1) as discussed in Section 3.4.3. Ultimately, in terms of expressivity, TAO is similar to Hontology as they both share ALCQU features, and to Acco as they share ALCU features; TAO does not have the H feature because it does not include role hierarchies (SubPropertyOf) as Hontology and Acco do; however, TAO has the IS features, because it defines inverse and transitive roles (relations), which the other two ontologies lack.

The indicators in Tables 3.1, 3.2 and 3.3 can be used to evaluate and compare TAO, Hontology, and the Accommodation Ontology based on their *transparency*, *flexibility*, and *cognitive ergonomics* [133]. *Transparency* is described as "the ability of an ontology to be thoroughly analyzed, with a detailed formalization of conceptual choices and motivation". *Flexibility* pertains to the ease of modifying and evolving the ontology with minimal side effects. Lastly, *cognitive ergonomics* refers to the ontology's capability to be "easily understood, manipulated, and exploited by end users". We will now discuss the main indicators of these properties.

implications of an OWL 2 ontology. See `https://www.w3.org/TR/owl2-syntax/#Entity_Declarations_and_Typing`

TAO performs favorably according to several transparency indicators [133], as it provides:

- a relatively *high number of axioms per class* (6.578). This is higher than Hontology, and lower than the Accommodation Ontology, mainly because the latter has a considerably smaller number of classes;
- a *low coupling* with external ontologies (0.03), similar to Hontology (0) and the Accommodation Ontology (0.06). This is calculated normalizing the number of external classes defined in other ontologies (NoC) by the total number of classes. Low coupling makes it easier for users to inspect and comprehend an ontology.
- a *strong cohesion* (i.e., relatedness among classes) due to the shallow 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 indicates a more vertical structure, related to a more extensive coverage of the tourism domain. This is higher than both Hontology (0.961) and the Accommodation Ontology (0.742).

An indication of *adaptability* [133] is provided by the combination of *minimal coupling* and *high cohesion*.

In conclusion, TAO demonstrates numerous indicators typically linked with favorable *cognitive ergonomics*, such as:

- a relatively *low class/property ratio* (0.499), which is even lower than Hontology (0.706) and Accommodation Ontology (0.705);
- a *sub-class tree with shallow depth and breadth* as suggested by ADIT-LN (3.612), max depth (6), and average breadth (6.578);
- a relatively *low entanglement* (0.179 in a range from 0 to 1) indicating that the inheritance tree is not highly complex.

Table 3.4 presents some statistics about the current prototype of TKG, encompassing over 10M triples that describe 35K facilities and nearly 898K reviews.

Figure 3.7 illustrates the distribution of individuals based on classes. The most common classes are (i) `tao:NormRating` and `schema:UserReview` used to model reviews; (ii) `acco:AccommodationFeature`[87] that serves as a general class for amenities along with a specific class from `tao` (e.g., `tao:Kitchen`, `tao:Television`); (iii) the classes utilized to model an offer such as `gr:Offering`, `gr:TypeAndQuantity-Node`, and `gr:UnitPriceSpecification`; (iv) `tao:Accommodation`, `gr:Quantita-ti-veValue`, `gr:SomeItems`, and `acco:BedDetails` are the classes used to represent an accommodation; (v) `tao:LodgingDescription`, `tao:LodgingFacility` (and its sub-classes), `schema:PostalAdress`, and `tao:NormAggregateRating` that are employed 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 their exact type.

[87]`tao:LocationAmenity` is defined as an equivalent class to `acco:AccommodationFeature`.

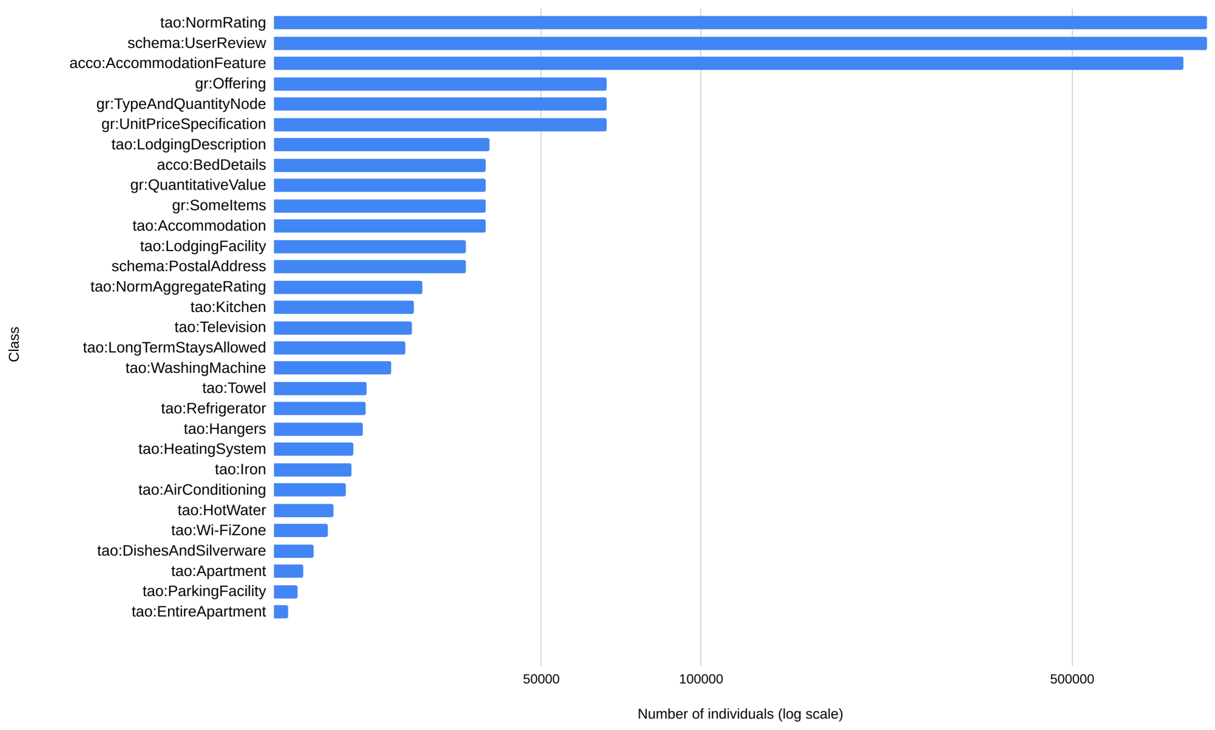


Figure 3.7: Top 30 classes by the number of individuals in the knowledge graph.

Table 3.4: Knowledge graph metrics

Metric	**Value**
Number of triples	10,917,081
Number of unique 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

3.7 Final Thoughts

In this chapter, we have introduced a framework for the semi-automated construction of a Tourism Knowledge Graph (TKG) used as a semantic layer above a corporate data lake, and introduced a novel ontology for representing this domain, known as the Tourism Analytics Ontology (TAO). We have assessed TKG and TAO based on functional, logical, and structural aspects. The methodology we adopted processes and incorporates data from a corporate data lake supplemented with public knowledge graphs (DBpedia and GeoNames).

The assessment indicates that TAO is i) more extended than the alternatives (Hontology and Accommodation Ontology) regarding the number of classes and axioms, and ii) also provides superior transparency, adaptability, and cognitive ergonomics.

In terms of the added value to the original data lake system, we can conclude that the semantic layer supplied by the knowledge graph offered numerous benefits to Linkalab's data lake platform: i) it manages data and metadata in a cohesive manner, ii) it possesses a flexible schema that can accommodate data variety and evolution, iii) it facilitates the development of algorithms and applications, and data science activities based on the data lake; iv) it integrates information in its graph structure that can be utilized by graph analytics [137, 138] and representation learning [139] algorithms; v) it includes knowledge extracted from texts alongside structured and semi-structured data commonly found in the data lake; vi) it can be employed to extend the data lake information context through connections with open knowledge graphs like DBpedia.

In future endeavors, we intend to pursue three main directions. Firstly, we are focusing on creating NLP solutions to enhance the extraction of entities from text, such as descriptions and reviews, to further enrich the representation of accommodation facilities. This includes the extraction of data from other sources related to various other tourist destinations employing solutions such as Entity Fishing[88] or Open Information

[88]`https://github.com/kermitt2/entity-fishing`

Extraction[89]

Secondly, we aim to develop a more scalable solution for maximizing the full data lake capabilities by integrating data about millions of facilities and users. To achieve this, we will depend on big data frameworks like 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 hasten the entity linking process.

Thirdly, we plan to develop an array of intelligent services based on TKG, starting from the accommodation offering optimization system presented in Chapter 5 but also including an entity-linking application for automatically annotating accommodations according to reviews and a conversational agent capable of answering queries about the tourism sector. In addition the application of graph completion techniques should allow us to predict relations between entities of the knowledge graph by leveraging Knowledge Graph Embedding models (e.g., TransE [140], RotatE [141], ComplexE [142]) or methods based on Graph Neural Networks [143], path-based features [144] and Few-Shot Learning [145].

In addition to these, we plan to expand the TAO ontology to model other aspects related to tourism, beginning with events and restaurants. We also intend to explore other APIs relevant to tourism such as Google Hotel API[90], Google Places API [91] or TripAdvisor[92] To conclude, we are focusing on automating as much as possible the pipeline we have used with the goal to create knowledge graphs with associated ontologies in any domain and sources.

[89]`https://openie.allenai.org/` can be utilized for named entity extraction, including entity detection, name resolution, and named entity recognition.

[90]See `https://developers.google.com/hotels`

[91]See `https://developers.google.com/maps/documentation/places/web-service`

[92]See `https://www.tripadvisor.com/developers` to reuse and extend TAO to also model their data in a unified manner.

Chapter 4

Improving Language Models with Knowledge Enhancement

The emergence of transformer-based language models in recent years has brought about significant advancements in the field of natural language processing, exhibiting outstanding capabilities across various areas. Yet, they still show considerable drawbacks. These deficiencies are particularly pronounced when handling intricate and highly specialized concepts or dealing with detailed factual data, especially in certain domains. In response to these issues and to further boost the efficacy of transformers within specific fields, scholars have shifted their focus towards knowledge enhancement. Broadly speaking, knowledge enhancement involves incorporating external knowledge into language models to bolster their performance in certain tasks. In this chapter, we deliver an extensive exploration of knowledge enhancement strategies for transformers. We direct this study towards the scholarly domain to yield general outcomes that can guide the implementation of analogous techniques within the tourism sector in Chapter 5. More specifically, within this chapter, we furnish a thorough summary and comparative analysis of four methodologies, assessing their effectiveness in categorizing scientific papers.

To serve this end, we devised a new benchmark comprising both 15K labeled articles and a knowledge graph consisting of 9.2K triples that depict relevant research subjects. A rigorous evaluation reveals that most of the introduced knowledge enhancement methodologies notably surpass the baseline set by Bidirectional Encoder Representations from Transformers.

4.1 Introduction

Over the past few years, models based on transformers have surfaced as potent instruments for tasks related to natural language processing, showcasing impressive outcomes across various fields. For example, BERT (Bidirectional Encoder Representations from Transformers) launched an innovative method that exploited bidirectional context, considerably enhancing the state of the art in multiple tasks, such as text classification, named entity recognition, sentiment analysis, and question answering [67]. More recently, GPT-4 [146] has shown extraordinary competence in producing coherent text and enabling more complex interactions between humans and machines [147].

However, transformers still face certain limitations. These deficiencies become particularly noticeable when dealing with intricate and complex concepts or with detailed factual information, specifically within certain domains [148, 149]. When we look at the scholarly domain, a critical task in this field is to effectively and accurately categorise scientific papers [150]. High-quality classification is essential in organising and retrieving knowledge, assisting researchers in keeping abreast of the latest developments and promoting the spread of information within the scientific community [151]. However, classifying scientific articles poses unique challenges for transformer models due to the complex language and nuanced domain-specific concepts found in scientific literature. As a result, transformers can have difficulty distinguishing between concepts that are quite distinct for domain experts, and in some situations, they may even create entirely fictitious information, a phenomenon known as hallucination [152]. Additional pre-training of existing transformers on specialised documents is a common technique to supplement a model with domain-specific knowledge [153, 154, 155, 156]. However, this method is quite demanding as it necessitates processing a large volume of domain-specific unlabelled text to adjust the model parameters effectively [157]. It also has limitations when it is necessary to use very specific vocabularies [157].

To address these issues and improve the performance of transformers in specific fields, researchers have shifted their focus to the concept of knowledge enhancement [158]. Knowledge enhancement involves incorporating external knowledge resources into the transformer models to augment their understanding and subsequently their performance in relevant tasks. Knowledge enhancement methodologies can manage many types of structured information. Notably, knowledge graphs (KGs) have emerged as powerful instruments for representing and organising structured data in a semantically meaningful way [159]. As we discussed in Chapter 2 and Chapter 3, KGs skillfully encapsulate the intricate relationships that exist between entities and attributes, providing a machine-readable representation of a domain for the benefit of various intelligent services [160, 161].

This chapter introduces a comprehensive study of knowledge enhancement approaches for transformers within the scholarly domain. Specifically, we offer a detailed review and comparative evaluation of four main methodologies, assessing their effectiveness in the task of classifying scientific articles.

To conduct this study, we created *AIDA15K*, a new benchmark for scientific article classification based on 15K scientific articles extracted from the Academia/Industry DynAmics Knowledge Graph (AIDA KG)[1] [162]. As external knowledge for the knowledge enhancement methodologies, we used the Computer Science Ontology (CSO) [163]. CSO is a large-scale ontology of research areas in the field of Computer Science. Compared to other solutions in this field (e.g., the ACM Computing Classification System), it offers a much more detailed representation of research concepts. For this reason, CSO was chosen by Springer Nature to automatically annotate proceeding books in Computer Science [151] and it is regularly used by a large number of tools to explore and analyse the scholarly domain [164, 165, 166, 167, 168].

We reveal several insightful findings about the impact of knowledge enhancement strategies on scientific text classification. Interestingly, even a basic method based on

[1]Academia/Industry DynAmics Knowledge Graph - `http://w3id.org/aida/`

directly appending domain knowledge to the text showed a significant improvement in performance. K-BERT [169], a more advanced version of this strategy, which controls the visibility of the injected knowledge to affect only relevant tokens, achieved significantly better results, especially with smaller training sets. The most effective strategy with larger training sets used a hybrid architecture, integrating BERT with a multilayer perceptron (MLP) to combine textual data with external knowledge [170].

The rest of this chapter is organised as follows. Section 4.2 gives a review of knowledge enhancement strategies. Section 4.3 defines the classification task under study and provides an overview of the BERT transformer, the AIDA Knowledge Graph, and the Computer Science Ontology. Section 4.4 describes in detail the new benchmark. Section 4.5 demonstrates four general methodologies for knowledge enhancement. In Section 4.6, we present and discuss the results of the comparative evaluation. Finally, Section 4.7 discusses the implications of our findings, highlights the limitations of the study, and outlines potential directions for future research.

4.2 Related Work

The advent of transformers has led the scientific community to recognize their shortcomings and consequently, prompt the development of Knowledge-Enhanced Pre-trained Transformers (KEPTs).

Due to the recent surge in research on KEPTs, the authors in [158] suggested a classification based on three characteristics: i) the granularity of knowledge, ii) the method of knowledge enhancement, and iii) the extent of symbolic knowledge parameterisation. In terms of the first characteristic, KEPTs incorporate knowledge at varying levels of granularity based on the task at hand. For instance, sentiment analysis tasks depend heavily on word features, requiring more information about individual entities, whereas text generation tasks depend on commonsense knowledge. With regard to the second characteristic, the method of knowledge enhancement is crucial in determining the effectiveness and efficiency of the integration between Pre-Trained Models (PTM) and infused knowledge, as well as for knowledge management and storage. Indeed, the technique used to infuse knowledge dictates what knowledge can be integrated and its form. In relation to the third characteristic, KEPTs are founded on the idea that symbolic knowledge can be utilized by PTMs in the form of semantic embeddings. To bridge the gap between symbolic knowledge and neural networks, the former is projected into a dense, low-dimensional semantic space and represented by distributed vectors using knowledge representation learning.

In this section, we are more concerned with examining the methods of knowledge enhancement, which [158] divided into six distinct strategies. *Feature-fused KEPTs*, which enhance the pre-training process by learning to represent additional features such as the sentiment polarity associated with a text (e.g., SentiLARE [171] and Ernie [172]), *Embedding-combined KEPTs* (e.g., Luke [173], Cokebert [174]), which merge a pre-trained model with KG embeddings created with representation learning algorithms by learning to weight the internal model embeddings with the infused ones using various architectures and customized pre-training objectives, *Data-structure-unified KEPTs* (e.g., K-BERT [169], K-LM [175], Colake [176], Comet [177]), which translate the relational triples of the KG into text and employ the same encoder for text and infused knowledge, *Knowledge-supervised*

KEPTs (e.g., Kepler [178] and ERICA [179]) that augment the PTM with relation information derived from a KG by additional pre-training tasks, *Retrieval-based KEPTs* (e.g., Realm [180] and [181]) that are mainly designed for question answering and utilize knowledge as an external reference, *Rule-guided KEPTs* (e.g., [182] and RuleBert [183]) strive to guide the PLM to resolve deductive reasoning tasks through fine-tuning or by using prompt tuning that incorporates prior knowledge into rules. However, many of these methods are inappropriate for the use case of this study, due to their need for pre-training the transformer on a large text corpus, establishing domain-specific rules, or querying external knowledge sources. We concentrate our focus here on methods that do not necessitate extensive pre-training and can be easily generalized across multiple domains. We impose this requirement to investigate knowledge enhancement in a low-budget / low-computation resource situation.

Specifically, we consider three broad strategies of knowledge enhancement.

The first strategy aligns with the data-structure-unified KEPTs as discussed in [158]. It aims to translate the relational triples from the knowledge graph into token sequences that are included in the input text. The primary benefit of this method is that the same encoder can be used to learn embeddings for both the text and the infused knowledge. As such, this solution circumvents the structural incompatibility issue between the pre-trained models and the knowledge-infused data. However, it should be noted that the creation of this unified data structure relies on the heuristic chosen for triple selection. A basic implementation of this technique may simply append specific triples to the text, whereas a more sophisticated method like K-BERT [169] strategically inserts triples within the text and manages their visibility, ensuring that only relevant tokens are affected. A potential disadvantage of this technique is that the transformation process eliminates the structural information inherent in the knowledge graph, which may limit the amount of context and relational data available for understanding intricate relationships in the data.

The second strategy, also commonplace in data-structure-unified KEPTs, entails conducting a lightweight pre-training [58] on a version of the knowledge base that has been translated into text [176, 184]. The effectiveness of this technique heavily depends on the nature and size of the knowledge base.

The third strategy, similar to what is done in Retrieval-based KEPTs, involves incorporating the knowledge as additional feature data during the training and inference of the classifier. These features can be quantitative, such as the number of authors of a document [170], or encoded as the embeddings of specific entities [173, 174]. In this context, the embedding vectors produced by the pre-trained model are combined with integrative features representing the additional symbolic or factual knowledge to enhance its ability to solve specific tasks.

4.3 Background

The present research compares a variety of strategies for augmenting knowledge in transformers, and their effects on enhancing the efficacy of text classifiers. The primary focus is on amplifying a BERT-based model for the classification of scholarly articles. Indeed, BERT [67] is amongst the most frequently employed pre-trained language models equipped with Transformer technology for deriving the context-sensitive representation of input text.

The classification of scientific papers is essential for the efficient distribution of academic literature, a task commonly performed by digital libraries, publishers, and analytics platforms [150, 185].

This job entails training a classifier to tackle a single-label multi-class classification challenge in which the classifier assigns the primary pertinent research field to each paper based on the paper's title and abstract. Put in more formal terms, given an array x containing n input samples and a set of labels l, the goal of model f is to assign each $x[i]$ to one of the labels in l. That is, the task is to compute $f(x[i]) = c$, where $c = 1, \ldots, l$ and for $i = 1, \ldots, n$.

To train and assess pertinent classifiers, we developed a benchmark consisting of 15K scientific papers equally divided among three research fields: Artificial Intelligence (AI), Software Engineering (SE), and Human-Computer Interaction (HCI). This benchmark also incorporates a knowledge graph of relevant research topics derived from CSO. The foundational idea of this resource is to use the knowledge graph as a tool to infuse additional knowledge into the classification process, with the ultimate aim of boosting the overall performance.

In the subsequent sections, we will provide an overview of BERT and the foundational data utilized for creating the benchmark: the AIDA Knowledge Graph and CSO.

4.3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model grounded on the Transformer framework discussed in Section 2.3, and it consistently delivers high-quality results across a variety of natural language processing (NLP) tasks. In this study, we applied BERT to text classification as outlined in Section 2.3.6.

4.3.2 The Computer Science Ontology

The Computer Science Ontology (CSO) is an extensive, automatically generated ontology of research areas. It encompasses approximately 14K topics and 159K semantic relationships, making it a thorough taxonomy of research areas in Computer Science. It was created by applying the Klink-2 algorithm [186] on a dataset of 16M academic articles.

CSO organizes research topics in a poly-hierarchical manner. While 'Computer Science' is the main root of this hierarchy, it also includes several other fundamental categories such as Linguistics, Geometry and Semantics.

CSO comprises three primary semantic relationships:

1. `superTopicOf`: signifies that a topic is a super-area of another topic. It represents a hierarchical relationship in which a broader concept includes a narrower concept. For instance, "software engineering" is a super topic of "software design".

2. `relatedEquivalent`: implies that two topics can be considered equivalent for the purpose of exploring research data. For example, "haptic device" is equivalent to "haptic interface".

3. `preferentialEquivalent`: indicates the primary label for topics that belong to a group of relatedEquivalent. For instance, both "ontology" and "ontologies" will have "ontology" as `preferentialEquivalent`.

Additional relationships available in CSO include `contributesTo`, which signifies that the research output of one topic contributes to another topic, and `sameAs` which maps research topics to similar entities in other knowledge graphs such as DBpedia[2], Wikidata[3], YAGO[4], and Cyc[5].

CSO is freely accessible and can be downloaded in various RDF formats (NT, TTL, XML) from the CSO Portal[6].

Compared to other existing methods (such as the ACM Computing Classification System), CSO covers a significantly larger number of research topics, enabling a detailed description of the content of academic papers. Indeed, CSO has demonstrated versatility in effectively supporting a wide variety of tasks, such as exploring and analysing scholarly data (e.g., ConceptScope [165], ScholarLensViz [164], Rexplore [187]), inspecting scholarly data with conversational agents (e.g., AIDA-Bot [188]), detecting research communities (e.g., ACE [189],), improved retrieval of research documents (e.g., CDSS [190]), identifying domain experts (e.g., VeTo [166]), refining the selection of keywords (e.g., R-Classify [191], ASM [192]), recommending articles [193] and video lessons [194], expanding existing ontology models [195, 196], generating knowledge graphs (e.g., Temporal KG [197], AIDA KG [162], CS KG [160], KGs for education [198]), knowledge graph embeddings (e.g., Trans4E [199]), topic models (e.g., CoCoNoW [167]), and analysing the impact of research teams (e.g., [200]).

Lastly, Springer Nature, a leading global research publisher, has also incorporated CSO into several of its innovative applications, such as i) Smart Topic Miner [151], a tool that assists the Springer Nature editorial team in categorizing proceedings, ii) Smart Book Recommender [193], an ontology-based recommender system for selecting books to promote at academic events, and iii) the AIDA Dashboard [201], a web application for exploring and analysing scientific venues, countries, and research topics.

4.3.3 Understanding the AIDA Knowledge Graph

The AIDA Knowledge Graph (AIDA KG) [162] is a comprehensive knowledge base that encompasses 21M scholarly articles and 8M patents. These records are enriched with diverse metadata, including the authors, institutions, nations, and venues. Additionally, they're associated with research fields from the Computer Science Ontology (CSO) [202] and corresponding industrial sectors from the Industrial Sectors Ontology (INDUSO)[7].

The creation of AIDA KG was facilitated by an automatic process that consolidates data from multiple sources, including OpenAlex, DBLP, the Research Organization Registry (ROR), DBpedia, and Wikidata. Its accessibility is public under the CC

[2]`https://www.dbpedia.org/`

[3]`https://www.wikidata.org`

[4]`https://yago-knowledge.org/`

[5]`https://cyc.com/`

[6]Download CSO from `https://w3id.org/cso`

[7]INDUSO - `https://aida.kmi.open.ac.uk/downloads/induso.ttl`

BY 4.0 license, and it can be acquired either as a dump or by querying via a triplestore `https://aida.kmi.open.ac.uk/sparql/`.

The architecture of this knowledge graph is based on the Resource Description Framework (RDF), discussed further in Section 2.

AIDA KG outlines eight categories of entities: scholarly articles, patents, authors, affiliations, journals, conferences, subjects, and industrial sectors. These entities are interlinked by 22 relationships, examples include: i) `hasAffiliation`, which denotes the affiliations of the authors, ii) `hasGridType`, which classifies the type of the affiliation as per the GRID classification (e.g., Education, Company, Government, Non-profit, and others), iii) `schema:creator`, which signifies the author of a scholarly article. A comprehensive list of relationships in AIDA KG can be found at `https://w3id.org/aida`.

4.4 Introduction to the AIDA15K Benchmark

This section provides a detailed overview of AIDA15K, our newly developed benchmark, which has been designed to assess and compare varying strategies for knowledge enhancement, specifically in the realm of categorizing scientific papers. The dataset is comprised of three main elements: i) a compilation of 15K labelled papers, each with a title and abstract, ii) a knowledge graph which outlines important research topics, and iii) additional data to aid in the application of knowledge enhancement methodologies.

4.4.1 Characteristics of the Scientific Papers

Determining the specific research field of a paper is a complex task, even for those who are experts in the field. Given this, we have chosen to use a labelling process that is based on the venue where the paper was published. To facilitate the process, we curated a diverse selection of journal and conference papers across three disciplines: Artificial Intelligence (AI), Software Engineering (SE), and Human-Computer Interaction (HCI). For instance, we included papers published at the Neural Information Processing Systems (NeurIPS) conference and the Nature Machine Intelligence Journal in the AI category.

The articles used in this study were extracted from the AIDA Knowledge Graph, and were selected based on their being published in the aforementioned venues post-2010 and having garnered at least 3 citations. To ensure a balanced dataset, we chose a random subset of 5k papers from each category, resulting in a total of 15K papers. Each article in AIDA15K is represented by an ID, in addition to its corresponding title and abstract.

4.4.2 Description of the Knowledge Graph

In order to strengthen our knowledge enhancement process, we created a knowledge graph containing 4,629 topics from the CSO that are relevant to the fields of AI, SE, and HCI. Every topic was matched with two statements extracted from the CSO ontology, leading to a total of 9,258 triples. These statements were selected by ranking all triples that had the topic as their subject, based on their predicates. The ranking prioritized ‘subTopicOf’, ‘superTopicOf’ (the inverse of ‘subTopicOf, materialized for this purpose), ‘preferentialEquivalent', and 'relatedEquivalent'. We then picked the top two instances from this ranking for each concept.

To illustrate, here is the description of the concept *image retrieval*, as it is defined in the CSO:

```
<image retrieval, subTopicOf, pattern recognition>
<image retrieval, superTopicOf, color and texture features>
```

4.4.3 Further Information Provided

The additional data supplied helps guide the methodologies for knowledge enhancement in selecting the relevant sections of the KG for a particular article. Our aim in including this data was to ensure a consistent and fair comparison of differing methodologies. In doing so, we assure that all methodologies involved in the evaluation have access to the same set of data, thus eliminating any potential variations due to differing implementation methods. Specifically, we provide 1) a mapping between each paper and the CSO topics it is related to, and 2) a specificity score for each topic.

The mapping was included because knowledge enhancement strategies often depend on entity-linking techniques to identify the most relevant sections of knowledge that pertain to the item in question [169]. Entity-linking techniques are used to link sections of text to the corresponding entities within a knowledge base [203]. As such, we applied the CSO Classifier [204] to all abstracts to discern which terms were associated with research topics in the CSO. The CSO Classifier is an unsupervised entity linking approach that uses a combination of string and word embeddings similarity to identify concepts described in the CSO. For instance, the paper with ID 4[8] is linked with six topics, such as *natural language processing*, *online learning environment*, and *recurrent neural networks*. This information will be utilized by the knowledge enhancement methodologies discussed in the next section to select the most relevant concepts and triples for a specific article.

The second piece of additional information, the specificity score, is crucial in optimizing the knowledge enhancement process, taking into account various constraints that limit the amount of information that can be incorporated when classifying a specific item. These constraints may be influenced by factors such as the 512-token input limitation inherent to the BERT model, as well as specific restrictions of the methodology itself. For example, the standard implementation of K-BERT incorporates only two triples for each entity recognized in the text.

In the AIDA15K dataset, papers are associated with an average of 7 topics, ranging from a minimum of 1 topic to a maximum of 35 topics. Many of the enhancement methods discussed in this work cannot handle the full range of topics associated with a paper. Therefore, they require the adoption of prioritization criteria to determine which topics are the most significant and, thus, should be considered and which ones should be omitted.

To assist in this process and align all the methodologies, we calculate the specificity score ss_i of topic i, which indicates its discriminative power with respect to the classification task. For this, we used Eq. 4.1, which calculates the maximum number of times a given entity e_i is found within the abstract of the papers associated with different research

[8] `https://aclanthology.org/2020.bea-1.13.pdf`

fields e_i^F, with $F = \{AI, SE, HCI\}$, and divides it by the number of times the same entity has been found within the whole AIDA15K train benchmark $e_i^{AI} + e_i^{SE} + e_i^{HCI}$.

$$ss_i = \frac{\max\left\{e_i^{AI}, e_i^{SE}, e_i^{HCI}\right\}}{e_i^{AI} + e_i^{SE} + e_i^{HCI}} \tag{4.1}$$

For example, if $ss_i = 0.33$, the topic i is associated with an equal probability with all the research fields and is thus unspecific. If $ss_i = 0.9$, the topic i is predominantly associated with just one research field and is thus highly specific.

4.5 Methods for Enhancing Knowledge

This section provides an overview of the four methods utilized to enhance knowledge.

Direct Text Injection. Relevant data can be seamlessly incorporated into the text, similar to the concept of prompt extension [205]. This approach adds pertinent triples to the end of the abstracts. Specifically, for each entity linked to the paper, two relevant triples are added. Before injection, the triples are altered by translating semantic relationships (e.g., *subTopicOf*) into English phrases ("is a narrower concept than"). The resulting sentences are integrated into the text.

K-BERT. K-BERT [169] is a widely used method for knowledge injection that enhances the text with triples from a Knowledge Graph (KG). It includes a Knowledge Layer that identifies the KG entities in the input text[9] and appends to them relevant triples, creating a "sentence tree". The sentence tree is processed by the Embedding Layer, assigning positional embeddings, and the Seeing Layer, filtering noise via the *visible matrix*. This matrix ensures that the injected predicates and objects only influence the embeddings of the entities they were attached to. The output from the previous layers is then processed by the Mask-Transformer Encoder, which adjusts the self-attention mechanism to accommodate the visible matrix. K-BERT has demonstrated superior performance to BERT in specific domains like finance, medicine, and law [169].

To apply K-BERT for our case, we modified its implementation[10] to handle English texts and integrate the KG outlined in Section 4.4.2. We tweaked the Knowledge Layer to recognize CSO topics' surface form in each sentence and append pertinent ontology triples.

Incorporating Extra Features via a Multilayer Perceptron. Instead of directly injecting the knowledge into the text, it can be introduced during the model fine-tuning and the classification process. For our case, we adopted this strategy by devising a neural network architecture that extends BERT using a multilayer perceptron (MLP). This approach has been employed in Chapter 5.

This method utilizes the BERT model to process the text, concatenates the resulting embeddings with additional features derived from relevant external data, and feeds them into the MLP. To suit our needs, we employed a standard English BERT model and added a component for generating a vector of features from the KG of research topics. For each article, the three most specific topics were selected and concatenated as text. We then

[9]K-BERT uses string match to identify entity labels in the text.

[10]Available at `https://github.com/autoliuweijie/K-BERT`

used Sentence BERT (SBERT) [206] to convert the resulting string into an embedding[11]. Finally, we concatenated this vector with the original text's embedding and fed it to the MLP, adding a final SoftMax layer to yield an output probability for each category.

Domain-specific Pre-training. Additional pre-training of BERT on a specific domain can be considered a form of knowledge injection [16]. It involves masking certain input tokens and requiring BERT to predict them based on the surrounding context. The best results are often achieved by masking about 15% of input tokens. We commenced with the standard english bert-base-uncased model[12] and extended its pre-training using text representations of CSO ontology triples as input.

4.6 Evaluation

This section offers a comparative assessment of the standard BERT with respect to the other BERT models that have been improved utilizing the four knowledge enhancement techniques we explored in Section 4.5. The models were evaluated using the benchmark outlined in Section 4.4.

4.6.1 Design of the Experiment

We took into account the following five methodologies presented in Section 4.5:

1. **BERT**, the uncased BERT model trained on textual features that we adopted as a baseline.
2. **BERT-DTI**: Direct Text Injection, which adds extra knowledge at the end of the input text.
3. **K-BERT**: Knowledge BERT, which augments recognized entities with suitable predicates and objects from the knowledge graph.
4. **BERT-MLP**: Integration of additional features utilizing a Multilayer Perceptron, the strategy that merges the BERT outputs with symbolic features.
5. **BERT-PT** Extra pre-training on the KG, which further pre-train BERT on a text generated by concatenating all triples in the KG.

For each experiment mentioned in this manuscript, we used 1500 documents for the development dataset and another 1500 documents for the test dataset. Each set contains 500 documents for each label (AI, SE, and HCI). By consistently using the same development and test datasets across all methodologies, we ensured that the performance outcomes are comparable.

We modified the size of the training datasets to evaluate the effect of different training sizes on model performance. The size of the training datasets was adjusted, with trials

[11]Sentence BERT is a method for producing fixed-length sentence embeddings that capture the semantic meaning of entire sentences.

[12]`https://huggingface.co/bert-base-uncased`

conducted using 3,000, 6,000, 9,000 and 12,000 articles, to assess the impact of different training sizes. In agreement with the findings reported in [207], we executed each configuration with ten different random seeds. The performances of the proposed methodologies have been measured by using the micro-average of the F1-score.

In all experiments, BERT was fine-tuned over five epochs. The training learning rate was set at 2×10^{-5}, the size of the sentence embedding vector was configured to 384, and the batch size was set to 6. The optimization method used was Adam, and the dropout probability was set at 0.1.

Train size	BERT		BERT-DTI		K-BERT		BERT-MLP		BERT-PT	
	avg	std	avg	std	avg	std	avg	std	avg	std
3000	0.855	0.008	0.858	0.009	**0.869**	0.009	0.850	0.010	0.850	0.002
6000	0.860	0.010	0.863	0.003	**0.871**	0.007	0.866	0.009	0.852	0.003
9000	0.868	0.005	0.868	0.005	0.870	0.006	**0.880**	0.019	0.865	0.005
12000	0.863	0.006	0.867	0.005	0.871	0.008	**0.880**	0.007	0.861	0.005

Table 4.1: F1-scores achieved for the five models at different sizes of the training set. The best results are highlighted in bold.

4.6.2 Results

As depicted in Table 4.1, the F1-score earned by various methods fluctuates depending on the dimensions of the training set.

The method **BERT-MLP** significantly outperforms the performance of the other methods for larger training sizes (9K, 12K). The method shows a notable increase over the BERT baseline, boosting the F1-score by 1.2% and 1.7% for the 9K and 12K training sets, correspondingly.

On the other hand, **K-BERT** manifests superior results for smaller training sizes (3k, 6k), notably securing a 1.1% F1-score enhancement over **BERT** with the 6k dataset. Nevertheless, in contrast to **BERT-MLP**, **K-BERT** does not appear to gain from larger training sizes, and its advantage over **BERT** diminishes as the dataset size expands.

The method **BERT-DTI** presents marginal improvements over the standard BERT. This indicates that even basic methods that incorporate knowledge into the text can be beneficial. However, it seems that more elaborate strategies produce significantly superior results.

Lastly, **BERT-PT** falls short of showing any significant enhancement over the BERT baseline. This might be attributed to the restricted size of the knowledge base used for pre-training. The outcome suggests that for similar situations, conventional knowledge enhancement techniques might be more advantageous than pretraining the model on domain-specific data.

In a nutshell, both **BERT-MLP** and **K-BERT** emerge as feasible choices for this task, but they exhibit very different trends. While K-BERT performs well with smaller training sets, its performance decreases as the size of the training set grows. In contrast, BERT-MLP performs excellently with larger training sets.

4.7 Conclusions

In this segment, we have carried out a comprehensive analysis of several knowledge enhancement approaches designed for transformer models in the scope of academic literature. This investigation provides a detailed summary and a contrastive assessment of four methods, emphasizing their proficiency in classifying scientific papers.

As part of this analysis, we have fashioned AIDA15k, a new benchmark that includes 15,000 academic articles derived from the AIDA Knowledge Graph. This benchmark also integrates a knowledge graph of 4,629 academic subjects, drawn from the Computer Science Ontology, aimed at providing additional knowledge for the sorting tasks. This resource's principal objective is to assess the overall efficiency of the different knowledge enhancement methods.

The relative evaluation on AIDA15k suggests that both BERT-MLP and K-BERT are suitable options for this assignment. Nevertheless, BERT-MLP is more compatible with larger training batches, whereas K-BERT outperforms with smaller ones. In addition, the results highlight that even basic knowledge enhancement methods, like affixing metadata to the input text, can yield positive outcomes.

In Chapter 5 we implemented knowledge enhancement methods in the tourism sector and in future studies, we plan to thoroughly examine other methodologies and apply all the strategies across various domains and tasks that require specialized knowledge. Our goal is to identify the best strategies designed for particular applications. This review will cover the use of these methods in crucial fields, such as academic research, travel, and news analysis. Also, we plan to conduct a thorough examination concerning the best representation of knowledge, also considering the trade-offs between small and concise representations and larger, potentially noise-ridden ones. This will entail a deep evaluation of how these representations affect the performance of LLMs, facilitating a more profound comprehension of the correlation between knowledge structuring and model efficiency.

Chapter 5

Optimizing Accommodation Offerings

Web-based platforms have emerged as the go-to solution for tourists to explore, compare, and book lodging for their journeys. Therefore, it becomes critical for online platforms and revenue managers to understand the nature of these interactions in order to devise a compelling and competitive proposal. The growth in Natural Language Processing, particularly due to the creation of large language models, has made remarkable strides in understanding the complex subtleties of human language. Similarly, knowledge graphs have proven to be powerful tools in representing and arranging structured data. However, efficiently combining these two potent technologies is still a work in progress. This chapter introduces an innovative deep-learning approach that amalgamates large language models with industry-specific knowledge graphs for the classification of hospitality offers. Our system's primary goal is to aid revenue managers in two key areas: i) understanding the market placement of their lodging offers, taking into account elements such as pricing and availability, user feedback, and demand, and ii) enhancing the presentation and features of the offers themselves, aiming to boost their overall attractiveness. For this reason, we have constructed a domain knowledge graph that encompasses a wide range of information about lodgings, and incorporated specific feature engineering strategies to reinforce the data representation within a large language model. To assess the efficacy of our method, we carried out a comparative study against other techniques using four datasets related to lodging offers in London. The suggested approach yields impressive results, markedly surpassing other techniques.

5.1 Introduction

As we navigate the era of digital transformation, online platforms have emerged as the main conduit for travellers to explore, evaluate, and reserve travel accommodations. With a plethora of information readily available, pinpointing the most fitting option can be a daunting task for users. Additionally, travellers often have unique requirements, such as location, facilities, price range, and specific interests. As a result, online platforms and revenue managers[1] need to grasp these dynamics thoroughly to devise a competitive and enticing proposition [208].

The evolution of Natural Language Processing (NLP), particularly with the advent of large language models based on transformers, has shown substantial improvements in grasping the complex subtleties of human language [146]. Transformer-based models have showcased impressive performance in various natural language comprehension tasks, including language generation, sentiment analysis, and question-answering [16, 209]. Concurrently, knowledge graphs have surfaced as powerful tools for structuring and organizing information in a semantic way [159]. These knowledge bases adeptly capture the relationships between entities and attributes, offering a machine-friendly representation of the domain to an array of intelligent services [160]. Typically, knowledge graphs organize information based on a domain ontology [210], which formally outlines entity types and relationships while facilitating reasoning processes. They can also be incrementally enhanced through link prediction techniques, aiming to discover additional relationships between domain entities [199, 211].

However, the effective amalgamation of these two potent technologies remains an active challenge, leading to several captivating problems [212]. The primary hurdles revolve around efficiently combining information from unstructured and structured data sources, as well as suitably encoding knowledge graph data.

This chapter introduces KGE-BERT (Knowledge Graph Enhanced BERT), an innovative deep-learning methodology that blends large language models with domain knowledge graphs with the objective of classifying tourism offers. Our method utilizes transformer models to gain a thorough understanding of accommodation descriptions which are presented as unstructured texts. This acquired knowledge is then seamlessly integrated with a detailed representation of the tourism domain derived from a knowledge graph that we constructed using Airbnb data, enhancing the system's classification performance. The underlying knowledge graph (Tourism Knowledge Graph - London) outlines over 65,000 accommodations modelled according to the Tourism Analytics Ontology (TAO) ontology[2] applying the methodology outlined in Chapter 3.

The primary goal of our system is to aid revenue managers in two key aspects: i) understanding the market positioning of their accommodation offerings, considering factors like price and availability along with user reviews and demand, and ii) optimizing their listings on online platforms. This optimization can be achieved by enhancing the style and level of detail provided in the description or by making modifications to the accommodations themselves. For example, adding new amenities that are generally associated with better reviews can boost the overall attractiveness of the offering. In particular, we concentrate on a variety of classification tasks that were identified through collaboration with Linkalab S.R.L, an Italian firm specializing in data science and data engineering[3], which has developed an industrial project on Tourism 4.0 called *Data Lake Turismo*[4] that gathers and analyzes tourism-related data from the web.

[1]In the tourism industry, the role of maximizing the performance of an accommodation is typically held by an individual known as a "Revenue Manager" or a "Revenue Optimization Manager".

[2]See `http://purl.org/tao/ns`

[3]Home page `https://www.linkalab.it/`

[4]"Turismo" is the Italian term for tourism.

When considering hospitality, business strategy must be informed by an array of information about consumer behaviour to maximize revenue and remain competitive. In terms of revenue management, it is crucial to devise specific strategies and adapt them to prevailing conditions. Hence, revenue managers require tools to analyze various factors such as pricing, availability, and distribution channels. Our system aims to provide additional support through the classification of an offer's positioning based on four predicted dimensions: 1) price range, 2) user interest, 3) relevance within a specific market, and 4) the customer's satisfaction after the stay. We will provide a detailed explanation of each relevant classification task in Section 5.4.

We assessed our approach by contrasting it with a BERT classifier and a baseline logistic regression classifier on a dataset of over 15,000 accommodation offers for each classification task. We also conducted a study on various combinations of feature types, highlighting their significant impact on the models' performance. The proposed solution achieves excellent results and substantially outperforms alternative methods, such as transformers models trained on the texts.

In conclusion, the main contributions of our work are as follows:

- We propose a novel methodology that effectively merges large language models and knowledge graphs within the context of the tourism domain.
- We provide an extensive evaluation demonstrating the advantages of our solution compared to traditional transformer models.
- We carry out a detailed analysis of feature engineering techniques to pinpoint the most effective combination of features.
- We share the complete codebase[5] of our methodology, which successfully addresses four classification tasks in the tourism domain.

This chapter is organized as follows. In Section 5.2, we present previous related work. Section 5.3 introduces the materials (e.g., ontology and knowledge graph) that our study considers. Section 5.4 discusses and formalizes the four tasks we address, while Section 5.5 provides a detailed description of the dataset used for training the machine learning models. Section 5.5 explores various strategies for feature engineering. Section 5.6 presents the architecture of our system and Section 5.7 reports the experimental evaluation. Section 5.8 discusses the results obtained. Finally, Section 5.9 discusses conclusions and potential avenues for future research.

5.2 Related Work

As we saw in Chapter 4, over the past few years, multiple efforts have been made to address the challenge of infusing structured knowledge into deep learning models for various tasks. For instance, [169] introduced K-BERT, an enhancement of BERT with a knowledge-enabled language representation model, where triples from knowledge graphs are infused into the sentences during the fine-tuning process. However, K-BERT does not

[5]`https://github.com/luca-secchi/kge-bert`

consistently outperform baselines like BERT for basic classification tasks. Additionally, [170] suggested a method for book genre classification that combines knowledge graph embeddings of authors with book titles and other metadata features to efficiently handle classification tasks. Xu et al. [213] presented a unique approach that incorporates entity-related knowledge into encoder-decoder large pre-trained language models (PLMs) through a generative knowledge infilling objective during continued pre-training. On the other hand, [214] suggested infusing domain-specific knowledge prior to fine-tuning task-oriented dialogue tasks using lightweight adapters. Finally, [215] outlined a method to incorporate structured knowledge into LLMs by directly training T5 models on factual triples from knowledge graphs (KGs). However, these methods generally focus on specific domains, require sizeable textual data and high budget / high computation resources for pre-training, or use domain-specific solutions and features that are not easily adaptable to the tasks tackled in this chapter.

Regarding the application of Knowledge Graphs to the tourism sector we already presented a detailed overview of previous experiences in Chapter 3. On the other hand, previous studies in peer-to-peer accommodation business (like Airbnb) were focused on the pricing issues [216] or in detecting the booking likelihood [217]. However, these methods do not consider the textual description of each accommodation and focus on numeric features, missing an important factor in the tourist's choices. Consequently, we introduced an approach that can also process textual descriptions leveraging the combination of knowledge graphs and Language Models.

Our proposed methodology integrates the benefits of previous approaches by incorporating knowledge graph information using an efficient feature engineering strategy with the aim of enhancing the accuracy of four tasks related to revenue management within the tourism sector. As far as we are aware, there have been no previous attempts to combine knowledge graphs and language models to solve classification tasks within the tourism sector.

5.3 Resources

This section outlines the resources utilized for constructing the classification system aimed at enhancing accommodation offers: the Tourism Analytics Ontology (TAO), the Tourism Knowledge Graph (TKG), DBpedia, and BERT (Bidirectional Encoder Representations from Transformers). Additionally, we explore varying knowledge enhancement methods for transformers.

5.3.1 The Tourism Analytics Ontology

The Tourism Analytics Ontology (TAO), introduced in Chapter 3, is an ontology designed to capture the intricate dynamics of the tourism sector and aid intelligent services within this sphere. It models diverse aspects of tourism, such as lodging facilities (i.e., the actual locations where accommodations are found), accommodations (i.e., components of the lodging facility available for rent, like hotel rooms), amenities (i.e., extra service, feature, or facility offered to guests), accommodation lease out offerings, tourist destinations (i.e., a location fundamental to a tourist's decision to travel), tourist locations (i.e., geographical point or area of interest), and tourist reviews and ratings. TAO was defined using OWL - Web Ontology Language [9] by following a data-driven design methodology to aid the making of a knowledge graph focused on hospitality and tourist attractions. TAO is based on various other open ontologies, including *Accommodation Ontology*[6], *GoodRelations*[7], and *Schema.org*[8]. It also incorporates four taxonomies, structured as hierarchies of OWL classes, concerning (i) amenities, (ii) lodging facilities, (ii) accommodations, and (iv) tourist locations. TAO possesses the capability to model information from several commonly employed data sources within the tourism sector, such as Booking.com[9] and Airbnb[10].

5.3.2 Creation of a Tourism Knowledge Graph

Following the methodology described in Chapter 3, we created a Tourism Knowledge Graph (TKG) based on TAO, designed to store and analyze information about tourism destinations and to assist in the development of intelligent applications in partnership with Linkalab. The version showcased in this chapter and utilized to create the benchmark for the experiments concentrates on 69K accommodations in London.

The knowledge graph retains information in the form of a data graph where nodes symbolize actual objects (like accommodation, a tourism destination, or a tourist location) and edges signify specific relations among these entities or between an entity and its properties. The data graph is articulated in RDF, and is retained in a triple-store database and can be queried using SPARQL (SPARQL Protocol and RDF Query Language).

The TKG version used here encompasses Airbnb's London accommodations and was constructed by reusing open data from the Inside Airbnb project[11]. Inside Airbnb is a project whose mission is to offer data and advocacy about Airbnb's influence on residential communities globally. For our analysis, we employed the information about the accommodations and their hosts that was collected for London on the 10th of September 2022. The data comprises 69,351 records with 75 fields[12] and describes specific accommodations (termed listing in Airbnb) identified with unique listing ids. We can differentiate three types of information: the textual description of the accommodation (e.g., `description`), the categorical fields that describe the characteristic of accommodation and the amenities (e.g., `amenities`), and other fields that can be regarded as numbers or converted into numbers and that specify some attributes of the accommodation (e.g., `price`). The ini-

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

[7]`https://www.heppnetz.de/projects/goodrelations/`

[8]`https://schema.org/`

[9]Booking.com is an online travel company that specializes in providing details about accommodations and related offers like hotels, B&Bs, and other types of hospitality. The website also lets users share their thoughts in the form of reviews.

[10]Airbnb is an American peer-to-peer company that connects hosts who offer their accommodation spaces, such as apartments and rooms, with travelers who are looking for a place to stay. Its website provides details on accommodations, as well as user reviews.

[11]See `http://insideairbnb.com/about`

[12]A data dictionary outlining each field is available online at `https://docs.google.com/spreadsheets/d/1iWCNJcSutYqpULSQHlNyGInUvHg2BoUGoNRIGa6Szc4`

tial file undergoes a series of dedicated data pipelines to convert its content into triples, generating the knowledge graph. Each field is processed in a unique manner, depending on the type of information it contains. Specifically, the description field, which contains unstructured text, undergoes processing to identify semantic entities such as places, food, and amenities. These entities are then linked to DBpedia, as described in Section 5.3.3. Through the process of entity linking, new triples are generated, explicitly representing in the knowledge graph the previously unstructured information extracted from the text. The information in the categorical field `amenities` is also transformed by means of an ontology mapping process that substitutes the short text describing an amenity with the corresponding class in the TAO ontology. This process standardizes the way amenities are referred to and allows us to refer to concepts instead of names. For instance, we use the specific TAO class *Television* instead of terms such as like "tv", "television", "tv with standard cable", "hdtv", "screen tv".

Numeric columns are encoded in the KG by generating triples where the subject is the accommodation, the predicate is the property represented by the column (e.g., `price`) and the object is a literal value found in the column (e.g., 100).

5.3.3 DBpedia

DBpedia [7] is a highly respected and meticulously maintained knowledge graph frequently utilized to enrich textual data with structured data. It has been created through the extraction of structured data from Wikipedia and its subsequent interconnection with other knowledge graphs, resulting in a comprehensive network of interrelated entities and relationships. DBpedia Spotlight [218] has risen to prominence in recent years as a dependable entity linking tool across varied domains. DBpedia Spotlight utilizes NLP methodologies to detect entity mentions in texts and link them to their corresponding DBpedia resources. This tool is regularly employed to associate domain knowledge graphs from diverse fields with a shared base of concepts [162, 219].

In our framework, we employ DBpedia Spotlight to extract auxiliary information from Airbnb accommodation descriptions. The extracted data acts as supplementary features for our classifier, covering a range of aspects such as accommodation specifics, associated named entities, and other relevant facets suggested by the entities mentioned in the text.

We present an illustration, highlighting certain words in a description and following it with a list of the respective DBpedia entities' IRIs (Internationalized Resource Identifier)[13].

Description text: *Ideally located within a brief distance of* ***Hampton Court Palace, Bushy Park, Kempton Park, Sandown and Twickenham Stadium****. [...] Boasts a high-end kitchen,* ***underfloor heating*** *and luxurious en suite bathroom.*

Extracted linked entities from text: `dbp:Hampton_Court_Palace`[14],
`dbp:Bushy_Park`, `dbp:Kempton_Park_Racecourse`,
`dbp:Sandown_Park_Racecourse`, `dbp:Twickenham_Stadium`,
`dbp:Underfloor_heating`.

[13]Refer to `https://datatracker.ietf.org/doc/html/rfc3987`

[14]The prefix domain format is used for brevity, where `dbp:Enity_Name` corresponds to the IRI `http://dbpedia.org/resource/Entity_Name`

5.3.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a notable language model capable of capturing extensive contextual representations of words and sentences [16] based on Transformers technology introduced in Section 2.3. Text classification is one of the primary applications of BERT as was described in Section 2.3.6.

In our case we used the final hidden state `(h)` of the initial token `[CLS]` as the representation for the entire sequence. To compute the probability `(p)` of a label `(c)`, a pooling layer is incorporated which consists of a fully connected layer with a parameter matrix `(W)` and a `tanh` activation function. The output from the pooling layer is then input to a `sigmoid` classifier, which is positioned on top of BERT. The complete formula is:

$$p(c|h) = sigmoid(tanh(Wh))$$

The parameters of BERT and W are co-tuned by maximizing the log probability of the correct label.

5.4 Problem Statement

In our collaboration with Linkalab, we distinguished four categorization problems that could help enhance a lodging offer. These tasks were the result of interactions with stakeholders and revenue managers.

We have given them definitions and labels as follows:

Task 1 - *Price Value.* Predict whether the lodging could be deemed high-value.

Task 2 - *User Interest.* Predict if a particular accommodation would attract potential customers.

Task 3 - *Relevance.* Predict if the lodging offer is a strong competitor in the market.

Task 4 - *User Appreciation.* Predict if customers would appreciate the accommodation after utilizing it.

Each of these tasks is encoded as a binary classification task, which allows the results to be used as useful checklists for revenue managers. This strategy enables the users of our systems to experiment with a variety of options and observe the impact of their choices on the predicted dimensions.

The *price value* categorization task utilizes two labels: low and high. An accommodation is classified as low-valued if its single-night stay price is below the median of the price for Airbnb accommodations in the destination after outlier elimination [15]. On the contrary, if the accommodation's price is higher, it is labeled as high-value. This categorization aids users in understanding the pricing they should propose, whether it's a first-time proposal or a response to market changes.

[15]We remove excessively large prices. We first calculate the mean μ and the standard deviation σ for all prices p, then we remove all prices where $p \geq \mu + 2\sigma$.

For the *user interest* task, we utilize the number of reviews to evaluate user interest. Indeed, Airbnb's review system requires that users can only leave a review once they have actually stayed at the property. Hence, the number of reviews can be viewed as an indirect measure of the property's occupancy and, correspondingly, the volume of booking requests received by the host. Even if the user gives a negative evaluation after staying, it still implies that the user has booked and stayed initially, demonstrating a genuine interest in the offer. In this case, we have defined two classes: *not interesting* and *interesting*. Accommodations classified as not interesting have not received any reviews in the last year, while interesting accommodations have received at least one review in the same time period.

To assess the *relevance*, we looked at the availability calendar for each accommodation for the next 365 days. We defined two classes: *high-relevance* if there is at least one bookable date in the next year and *low-relevance* otherwise[16]. If an accommodation becomes unavailable for booking for extended periods (365 days), we speculate that this is because it's not competitive in the market, and this could be related to its offerings and how they're presented. Thus, this classification serves as a warning about the long-term relevance of the offer.

Finally, we assessed the *user appreciation* using the Average Review Score. On Airbnb, each user is required to provide six different review scores about specific aspects of their experience: accuracy, cleanliness, check-in, communication, location, and value. The Average Review Score is a number from 1 to 5 calculated as the average of these 6 scores. We defined two classes: *highly appreciated* if the average review score is greater than 4.5 and *normally appreciated* otherwise.

5.5 Data Sources

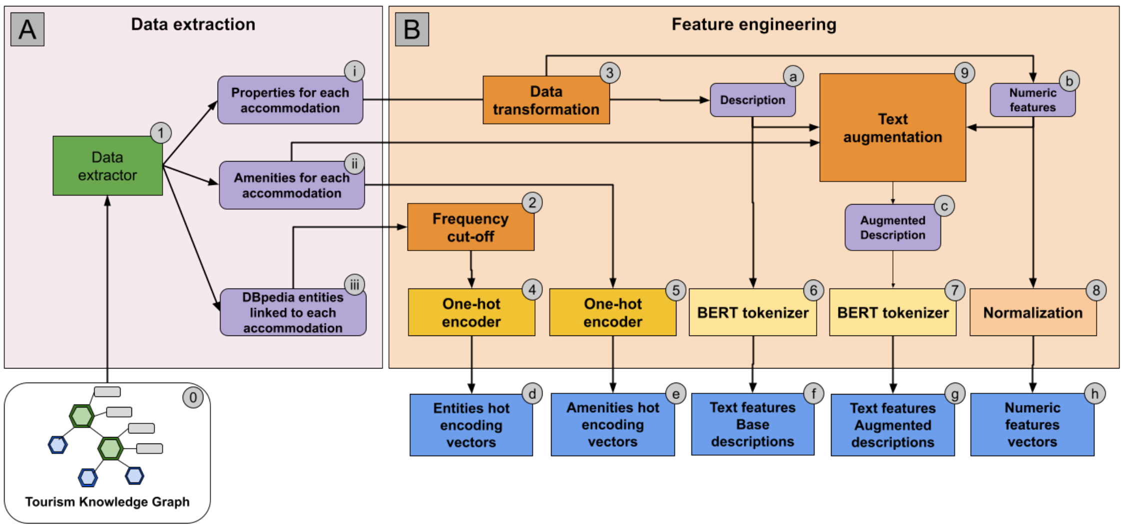
We employed TKG, described in Section 5.3, to generate all the data required to train and evaluate our model on the four categorization tasks. Since TKG is stored in a triple-store database[17], we were able to extract all pertinent data using the SPARQL language. The extraction process is depicted in the block labelled with A in Figure 5.1. The Data extractor process generates three distinct datasets (i, ii, iii in the diagram) that are then used for feature engineering: (i) a dataset where each accommodation is associated with its description text and with all its properties expressed as numbers, dates or true/false flags (e.g., number of rooms, first review date, instantly bookable flag); (ii) a dataset where each accommodation is associated with all the included amenities expressed with TAO classes; (iii) a dataset where each accommodation is associated with all related DBpedia entities expressed as IRIs.

We processed the datasets we obtained from TKG to generate four kind of features: (i) *textual features*, serving as an ideal input for transformer models like BERT, (ii) *numerical features*, (iii) *categorical features*, and (iv) *linked entities*, which comprise DBpedia entities

[16]While it's not possible to determine why an accommodation is unavailable for booking on a particular date, it's highly unlikely that it's booked for the entire next year.

[17]We used the free version of GraphDB `https://graphdb.ontotext.com/`. GraphDB is an enterprise-ready Semantic Graph Database, compliant with W3C Standards.

Figure 5.1: Knowledge graph data extraction and feature engineering process.



extracted from descriptions. This procedure is depicted in Block B of Figure 5.1. The procedure includes the steps below. Initially, property details (i) undergo a Data transformation process (3) to preprocess the textual descriptions (a), which are then converted into text features presented as tokens (f) by a BERT tokenizer (6). The same transformation also yields a numeric value vector (b) which is normalized by a specialized process (8) to generate a numeric feature vector (h). Amenities (ii) are converted into numeric vectors (e) using a one-hot encoding process (5). DBpedia entities (iii) are simplified to a manageable count by discarding those linked to fewer than 100 accommodations (2) and then converted into numeric vectors (d) using a one-hot encoding process (4). Lastly, we employ a text augmentation process (9) to create an enhanced version of the descriptions that also includes numeric features and amenities (c). This is processed by a BERT tokenizer (7) and converted into another text feature set (g) which will be used to test the ability of transformers to directly process structured features. The longest textual description in our dataset consists of 198 words, with the average length being 114 words. As a result, these descriptions can be conveniently processed by BERT, which can handle texts with a maximum token limit of 512. Furthermore, we have sufficient "space" within this limit to include additional features in text format.

The numerical features include a variety of metrics already in numeric format, such as the number of bedrooms, beds, bathrooms, minimum night stays, etc. All dates (for instance, the first review date) are stated as the number of days in the past relative to the day the original data was collected from Airbnb, in our case the 10th of September 2022. Finally, true/false flag values, like instantly bookable flags, are converted into numeric values of 1 or 0. By concatenating the numerical features, we create an n dimension vector, whose length will vary slightly for each task as we exclude the predicted variables specific to that task, as well as all correlated values. For instance, for the user interest classification task, we exclude all metrics about the number of reviews (e.g., number_of_reviews, first_review, last_review) or those related to review scores (e.g., review_scores_rating, review_scores_accuracy, etc.). Likewise, for the relevance classification task, we exclude all variables about availability (e.g., availability_30, availability_60, availability_90, availability_365). As the final step, we performed unity-based normalization on this vector, with the aim of scaling all values within the range of [0, 1]. This normalization process ensures a standardized representation of the vector, enabling comparisons and analysis across different variables.

5.5.1 Injecting Information as Text

To improve the classification process, transformers like BERT can be given additional information by expanding the description texts with injected knowledge in the form of key terms and numbers. This can be seen as a type of prompt addition as described by [205]. To evaluate this method, we produced the augmented descriptions (g) shown in Figure 5.1. We did this by adding numeric and categorical features after the accommodation description. Entities extracted from the description in the previous steps are excluded to avoid duplication.

More precisely, we used the following method: (i) numeric property values were included as text, with each value separated by spaces; (ii) TAO amenities were included by adding their corresponding labels, separated by spaces. Regarding numeric value injection as text, prior works [220, 221] have confirmed that BERT can handle numeric values in this form. Here is an example of an accommodation description text extended with the list of amenities (in bold) and the numerical features (in underlined text):

"This beautifully decorated two-bedroom serviced apartment is conveniently located in the vibrant Shoreditch area [...] ***dishes and silverware cable tv cooking basics bathtub carbon monoxide alarm smoke alarm heating lockbox first aid kit*** *[...] 1 2.0 3.0 4.0 [...]".*

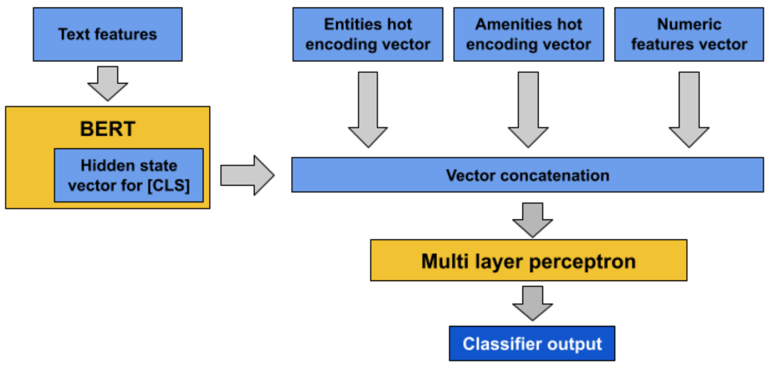
In this example, we list in bold ten amenities labels and, underlined, four numeric properties associated with the accommodation whose meaning is based on their position in the sequence. The latter are the host-is-super-host binary flag (value 1), the number of bedrooms (value 2.0), the number of beds (value 3.0), and the minimum nights bookable (value 4.0).

5.6 Architecture

We propose a method, hereafter referred to as KGE-BERT (Knowledge Graph Enhanced BERT), which integrates a Transformer model for textual data processing and a Multi-Layer Perceptron (MLP) for incorporating other feature types. Figure 5.2 portrays the architecture. The model accommodates knowledge enhancement from TKG by merging the four types of features discussed in Section 5.5: textual, numeric, categorical, and linked entities.

During the model training phase, we perform an end-to-end optimization process for each classification task. This process involves two distinct operations. Firstly, the BERT transformer is fine-tuned on the set of descriptions. Secondly, the MLP component is trained from the ground up on all the features. By merging these two operations, our

Figure 5.2: KGE-BERT model architecture.



method achieves a comprehensive training procedure that maximizes the potential of both the fine-tuned BERT transformer and the newly trained MLP.

The tokenized text is processed by BERT (using the English uncased model[18]). For BERT output, we use the hidden vector state associated with the first character of the input, represented by the `[CLS]` token. The `tanh` output of the pooling layer attached to BERT is scaled from 0 to 1 in order to match the range of other non-textual feature vectors. The numerical features are encoded as vectors of real numbers normalized from 0 to 1, while categorical and linked entities' features are expressed as hot encoding.

The resulting four vectors are concatenated and used as input for the MLP, which has two layers with 1024 units each and uses ReLu as the activation function. All the MLP layers undergo a dropout process with default probability *p=0.1* during the training phase to prevent over-fitting. The MLP output layer is a Sigmoid layer that provides the probability output for the classification.

5.7 Evaluation

To evaluate the reliability of our system, we mede a comparison with numerous benchmarks for the quartet of tasks outlined in Section 5.4.

5.7.1 Evaluation Procedure

For each of the four categorization tasks, we constructed a distinct balanced dataset, derived from the comprehensive datasets detailed in Section 5.5. We partitioned the balanced dataset for each task into three segments: training, validation, and test sets. Next, using the entire training set, we generated four unique training sets for each task, each escalating in size to include 3000, 6000, 9000, and 12000 accommodations. The purpose of this exercise was to investigate the impact of differing data sizes on the efficacy of the categorization tasks. This is particularly pertinent for real-world applications where assembling a sizable training set might present difficulties. For each task, we utilized the

[18]See Hugging Face repository `https://huggingface.co/bert-base-uncased`

same validation and test set, each with a fixed size of 1800 elements, to ensure comparable outcomes. We employed the validation sets to execute hyperparameter adjustment for the categorization threshold. Since the sigmoid layer of the model yields the probability $p(1)$ that the anticipated label is 1 and $p(0) = 1 - p(1)$ that the anticipated label is 0, we didn't use a static categorization threshold of 0.5 to determine the output class, but selected it with a hyper-parameter search using the validation dataset by scrutinizing all possible threshold values from 0.1 to 0.9 in increments of 0.1. The optimal threshold is then incorporated into the model during the final assessment of the test set.

We selected the macro-average of precision, recall, and F1 score as performance metrics. Specifically, we repeated each training and hyper-parameters tuning (utilizing the evaluation set) five times, generating 5 variations of the trained model for each experiment. Then we assessed each model variation on the test set and calculated the average value of each metric (macro-average F1-score, recall, and precision). Indeed, previous studies have demonstrated that when a BERT-based model is fine-tuned numerous times on the same dataset, with only the random seed being altered, it can cause considerable variation in accuracy [16, 222].

5.7.2 Methods

We conducted our experiments employing seven methods, which are also encapsulated in Table 2:

1. *LogisticRegression.* A basic Logistic Regression Classifier that we utilized as a benchmark. It was structured as a network with 1 concealed layer of 1 unit, ReLu activation functions, and a concluding Sigmoid layer to produce the binary categorization probability. Since it was incapable of processing text, it was furnished with only numerical, categorical, and linked entity features.

2. *BERT.* The BERT-based uncased model, trained on text features. The BERT model pooled output is supplied to a final inner layer with one unit and a concluding Sigmoid layer.

3. *BERT-injected.* A variant of the previous method that employs the technique for knowledge injection outlined in Section 5.5.1 for also integrating numerical, categorical, and linked entities features into the text.

4. *KGE-BERT-1hot.* KGE-BERT as explicated in Section 5.6, utilizing only accommodation descriptions and the one-hot encoding of the categorical entities from TAO and linked entities from DBpedia;

5. *KGE-BERT-num.* KGE-BERT, using solely accommodation descriptions and numerical features.

6. *KGE-BERT-full.* KGE-BERT, utilizing the entire set of features.

7. *KGE-BERT-injected-full.* KGE-BERT, utilizing the entire set of features and also infusing the textual descriptions with additional information as outlined in Section 5.5.1.

All BERT models were fine-tuned by modifying the original hyperparameters proposed in the seminal BERT paper [16]: the batch size is 8 (to accommodate our GPU limitations and to procure better outcomes during training), the learning rate is 2^{-5}, the optimization method is *Adam*, the dropout probability is 0.1, and the training epochs are 5.

Table 5.1: Descriptions of Experiments

Name of Experiment	Textual Features	Numerical Features	Linked Entities	Categorical
LOGISTIC REGRESSION	-	Numerical properties	DBPedia	TAO Amenities
BERT	Description	-	-	-
BERT-injected	Injected Description	-	-	-
KGE-BERT-num	Description	Numerical properties	-	-
KGE-BERT-1hot	Description	-	DBPedia	TAO Amenities
KGE-BERT-full	Description	Numerical properties	DBPedia	TAO Amenities
KGE-BERT-injected-full	Injected Description	Numerical properties	DBPedia	TAO Amenities

We conducted all experiments on a workstation with the following specifications: CPU INTEL CORE I9-7900X 3.3G (4.3G TURBO) 10CORE, GPU Nvidia GTX1080 TI 11GB VRAM, SSD 1TB, RAM 65GB.

5.7.3 Results

The subsequent four subsections will detail the conclusions drawn from each classification task. To improve comprehension, all macro-averages of F1-score, precision, and recall are displayed in percentage points.

Price value classification

The results are documented in Table 5.2 and visually represented in Figure 5.3. KGE-BERT-full surpasses all other methods, showcasing its capability to combine different features. KGE-BERT-injected-full comes second, indicating that infusing features both as vectors and in the text occasionally hinders performance. Nonetheless, BERT-injected fares better than BERT, underlining the importance of integrating a collection of diverse features into the model.

Interestingly, the employment of solely numerical features (KGE-BERT-num) appears to have a detrimental impact on performance for this task.

Relevance classification

Similar to the previous task, the results for this task can be found in Table 5.3 and in Figure 5.4. KGE-BERT-injected-full acquires the highest results here, proving that infusing features both as vectors and in the text can enhance performance. For this task, BERT underperforms (`F1-score <67%`), and it is even surpassed by the LinearRegression baseline. This implies that descriptions of accommodations may not be the best indicators of relevance. On the contrary, numerical features seem to be the most impactful. Indeed, KGE-BERT-num attains exceptional results.

Figure 5.3: F1-value macro-average results for each experiment in Value classification task.

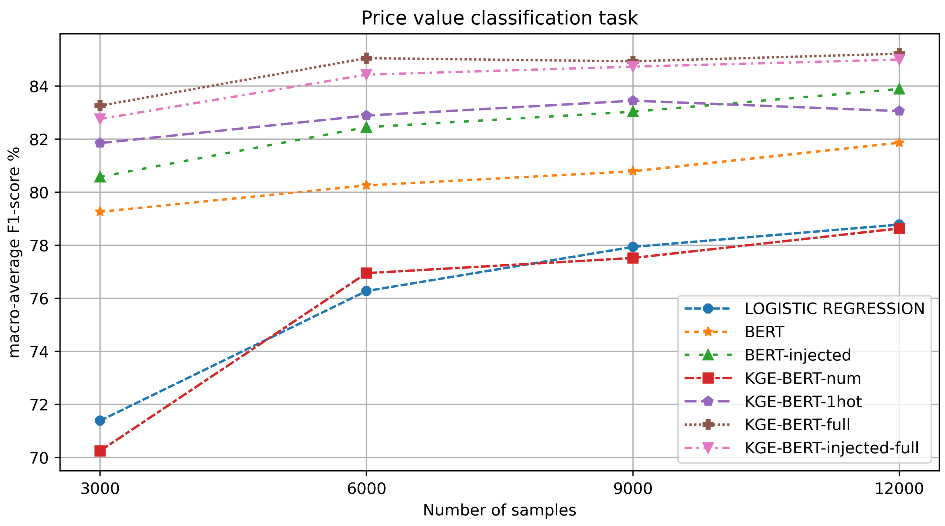


Table 5.2: Price value classification task: macro-average values for F1-score, precision and recall results.

				f1-score %				precision %				recall %
max_train_size	3000	6000	9000	12000	3000	6000	9000	12000	3000	6000	9000	12000
experiment												
LOGISTIC REGRESSION	71.38	76.27	77.94	78.78	71.94	76.65	79.18	79.34	71.50	76.32	78.10	78.84
BERT	79.26	80.25	80.79	81.86	79.73	80.81	80.94	81.28	78.97	79.87	80.72	82.62
BERT-injected	80.58	82.45	83.03	83.89	80.56	82.29	82.61	84.04	80.73	82.70	83.53	83.89
KGE-BERT-num	70.24	76.95	77.52	78.63	71.28	77.42	77.64	78.71	70.51	77.02	77.55	78.64
KGE-BERT-1hot	81.85	82.88	83.45	83.05	81.99	83.05	83.57	83.19	81.86	82.89	83.45	83.06
KGE-BERT-full	**83.26**	**85.05**	**84.93**	**85.22**	**83.28**	**85.25**	**85.03**	**85.25**	**83.26**	**85.06**	**84.93**	**85.22**
KGE-BERT-injected-full	82.76	84.43	84.73	85.00	82.90	84.55	84.81	85.18	82.76	84.43	84.73	85.00

Table 5.3: Relevance classification task: macro-average values for F1-score, precision and recall results.

				f1-score %				precision %				recall %
max_train_size	3000	6000	9000	12000	3000	6000	9000	12000	3000	6000	9000	12000
experiment												
LOGISTIC REGRESSION	66.09	68.63	68.82	69.92	66.98	69.11	68.82	70.01	66.33	68.83	68.83	69.98
BERT	58.96	64.63	65.77	66.34	62.73	66.69	65.87	66.27	61.30	63.56	66.12	66.81
BERT-injected	77.25	78.07	79.14	79.68	78.16	78.31	79.34	80.25	76.93	78.44	79.31	79.87
KGE-BERT-num	77.44	**79.99**	81.35	81.41	77.89	**80.57**	82.16	82.18	77.60	**80.18**	81.57	81.63
KGE-BERT-1hot	71.56	73.72	74.29	75.22	71.66	73.77	74.44	75.41	71.61	73.74	74.35	75.21
KGE-BERT-full	75.54	78.75	80.40	81.02	76.47	79.12	80.74	81.91	75.83	78.89	80.52	81.26
KGE-BERT-injected-full	**78.45**	79.51	**82.58**	**83.61**	**79.07**	80.08	**82.92**	**84.47**	**78.65**	79.69	**82.69**	**83.83**

Figure 5.4: F1-value macro-average results for each experiment in Relevance classification task.

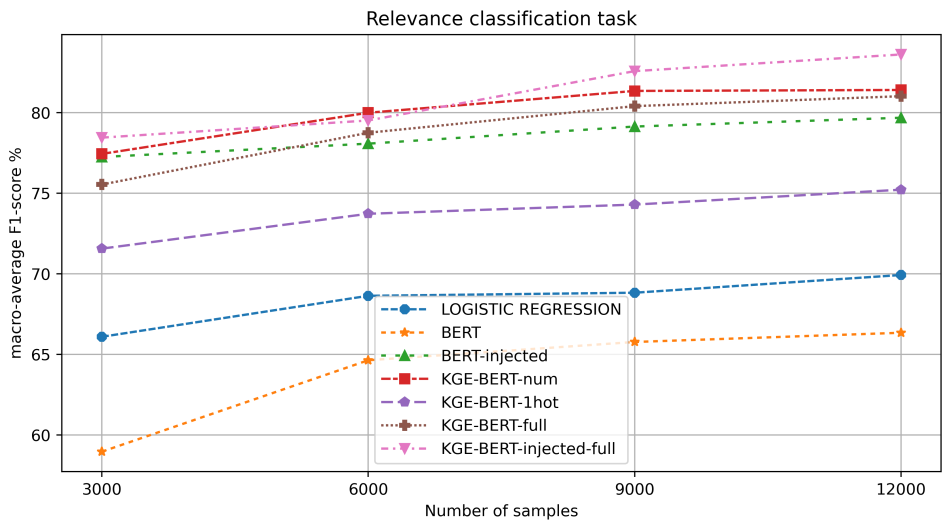


Table 5.4: User interest classification task: macro-average values for F1-score, precision and recall results.

	f1-score %				precision %				recall %			
max_train_size	3000	6000	9000	12000	3000	6000	9000	12000	3000	6000	9000	12000
experiment												
LOGISTIC REGRESSION	72.94	76.30	78.05	78.68	73.09	76.74	78.06	79.08	72.98	76.40	78.05	78.76
BERT	64.44	67.29	66.97	67.92	65.57	66.82	66.96	67.57	64.76	68.14	67.61	68.53
BERT-injected	81.33	82.50	83.26	83.91	81.78	82.56	82.80	83.77	81.24	82.61	83.90	84.26
KGE-BERT-num	83.63	83.34	83.66	83.65	83.85	83.69	83.94	84.07	83.67	83.40	83.71	83.71
KGE-BERT-1hot	80.79	80.82	82.15	82.11	80.82	80.92	82.16	82.12	80.80	80.85	82.15	82.11
KGE-BERT-full	85.16	**85.67**	85.63	86.02	85.45	**85.86**	86.10	86.32	85.21	**85.70**	85.70	86.06
KGE-BERT-injected-full	**86.05**	85.36	**87.09**	**88.22**	**86.92**	85.82	**87.89**	**88.65**	**86.15**	85.41	**87.18**	**88.27**

User interest classification

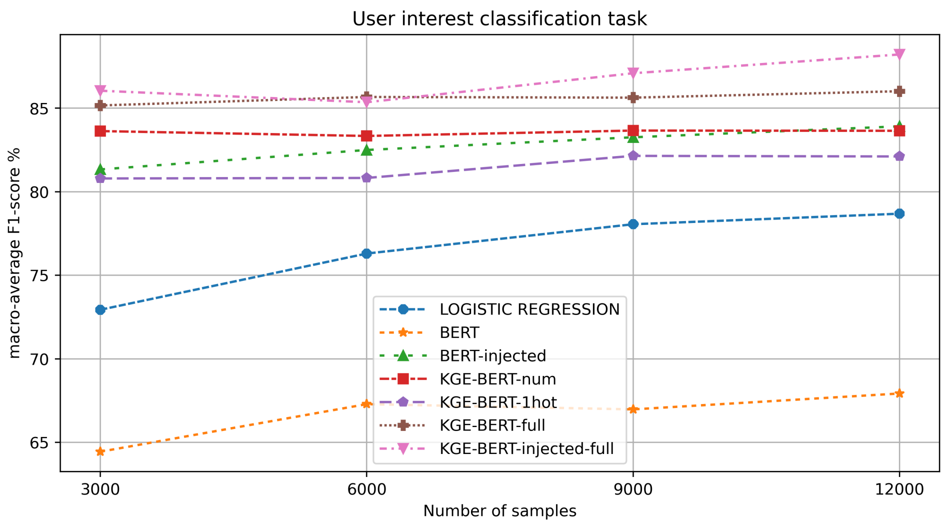
The results for this task are listed in Table 5.4 and in Figure 5.5. KGE-BERT-injected-full again attains the highest results, particularly in the experiments using a large number of training samples (9000 and 12000). KGE-BERT-full is a close competitor, achieving the highest F1 on 6000 training samples. As previously, BERT-injected surpasses BERT.

User appreciation classification

The results for this task are outlined in Table 5.5 and in Figure 5.6. This appears to be the most difficult amongst the four classification tasks as indicated by the comparatively low *F1-score*. This is because user reviews are submitted post-visit and are influenced by the user's actual experience, which is challenging to predict based on the accommodation information available to the model.

KGE-BERT-injected-full and KGE-BERT-full achieve similar results, significantly out-

Figure 5.5: F1-value macro-average results for each experiment in User interest classification task.



doing all the other methods. The other variants of KGE-BERT also perform better than BERT and are comparable to BERT-injected. This once again suggests that the amalgamation of multiple features is beneficial.

Figure 5.6: F1-value macro-average results for each experiment in User appreciation classification task.

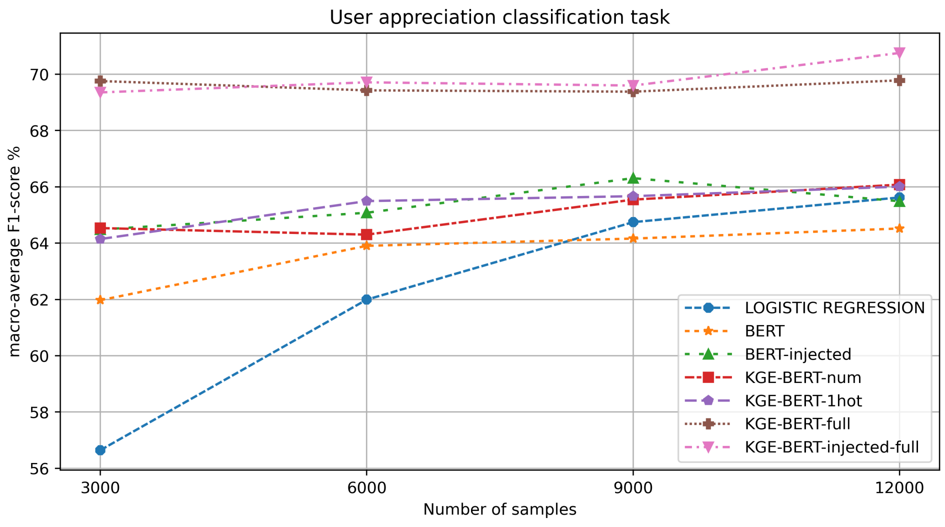


Table 5.5: User appreciation classification task: macro-average values for F1-score, precision and recall results.

	f1-score %				precision %				recall %			
max_train_size	3000	6000	9000	12000	3000	6000	9000	12000	3000	6000	9000	12000
experiment												
LOGISTIC REGRESSION	56.64	61.99	64.75	65.62	56.67	62.00	64.76	65.72	56.65	62.00	64.75	65.64
BERT	61.97	63.90	64.16	64.52	63.07	63.90	64.44	64.81	61.64	64.61	64.09	64.70
BERT-injected	64.48	65.08	66.31	65.49	64.35	64.53	66.46	65.63	65.50	65.83	66.56	65.66
KGE-BERT-num	64.54	64.30	65.55	66.08	64.67	64.36	65.61	66.32	64.60	64.32	65.56	66.17
KGE-BERT-1hot	64.15	65.49	65.67	66.01	64.95	65.98	66.48	66.38	64.40	65.63	65.90	66.11
KGE-BERT-full	**69.76**	69.43	69.38	69.78	**70.47**	70.67	**70.98**	71.51	**69.90**	69.70	69.74	70.16
KGE-BERT-injected-full	69.36	**69.71**	**69.60**	**70.76**	70.15	**70.84**	70.96	**71.58**	69.53	**69.95**	**69.90**	**70.92**

Table 5.6: ΔF1 - increments of F1-score for BERT-KG-injected-full compared to BERT.

max_train_size	3000	6000	9000	12000
Relevance classification task	19.49	14.88	16.81	17.27
User interest classification task	21.61	18.07	20.12	20.29
Price value classification task	3.50	4.18	3.94	3.13
User appreciation classification task	7.38	5.81	5.44	6.24

5.8 Analysis

Our experimental data suggests that the KGE-BERT methodology provides significant benefits in comparison to traditional transformer techniques. For all combinations tested, we observed superior *F1-score* metrics for the two KGE-BERT variants that leverage all accessible features (textual, numerical, categorical, and linked entities). KGE-BERT-injected-full and KGE-BERT-full yielded similar outcomes. The former method may be preferable when dealing with texts shorter than the maximum token limit of the transformer, allowing additional space for knowledge injection as text. The latter method is capable of handling longer texts and requires less computational resources during training and inference.

In summary, our experimental data underscores the benefits of amalgamating textual, categorical, and numerical features in this intricate domain.

To further emphasize the distinction between the conventional BERT and KGE-BERT-injected-full method (0), we mark in Table 5.6 the increase in *F1-score*. The values presented in the table are computed using the formula

$$\Delta F1 = F1_{KGE-BERT-injected-full} - F1_{BERT}$$

The recorded maximum gain illustrates a substantial enhancement of over 21 percentage points. In the context of the price value classification, where the BERT model showcases the finest performance, the integration of knowledge enhancement still results in a noticeable improvement of more than three percentage points. This observation suggests that there is potential for further improvement through knowledge enhancement, even when BERT performs optimally.

5.9 Final Remarks

In this study, we introduce a hybrid system that aids the optimisation of accommodation offers by integrating large language models and knowledge graphs in the tourism sector. Our methodology was applied to four distinct classification tasks that are fundamental to optimizing how an accommodation can be positioned in terms of price value, relevance, user interest and user appreciation. For this, we implemented a knowledge enhancement method that employs a knowledge graph to offer numerical data, categorical details, and linked entities to improve standard transformer models for classification. We also assessed an alternative strategy that reiterates the external knowledge also by injecting the same information directly as text. Our methodology using this solution (KGE-BERT-injected-full) evidently surpassed the standard BERT model, achieving a mean F1-score improvement of 11.7 percentage points (ranging from a minimum of 3.1 to a maximum of 20.6, depending on the task). When the combined enhancement strategy cannot be employed due to limits on the max number of tokens, an simpler enhancement approach can be utilized. This solution (KGE-BERT-full) also exceeded BERT with a mean F1-score growth of 11.0 (ranging from a minimum of 3.4 to a maximum of 20.7).

A fascinating direction for future research involves improving the system to support multi-class classification and regression, allowing for more nuanced results for each dimension. Another expansion of the current work could involve examining the extent to which a classifier trained for a specific tourist destination (e.g., London) can be utilized for another destination (e.g., Rome), and investigating the potential for transfer learning. In addition, we are exploring methods to provide explanations to users and offer customized recommendations on how to modify the descriptions and features of accommodations. To achieve this, we plan to employ explainable AI techniques [223] and generative language models [146], which present a promising approach. The ultimate aim is to provide transparent and interpretable insights, empowering users to make informed decisions and enhance the quality of accommodation offerings.

Chapter 6

Conclusions

The current age of big data has caused a paradigm shift in the World Wide Web, transforming it from a content-centric to a data-centric platform. Knowledge Graphs based on semantic web technologies are at the forefront of these changes, together with big data architectures like data lakes. In this scenario, the emergence of Artificial Intelligence in the form of deep learning became possible thanks to the vast amount of data now available for training increasingly powerful models. In this thesis we explored how all these converging factors can be combined to support the creation of intelligent applications within the tourism domain.

The first research question we tried to answer was how to guide the design of a new ontology with a data driven approach (Q1). To answer this question our first contribution is the definition of a general data-driven methodology for the semi-automatic generation of knowledge graphs, which we have applied to the tourism domain.

We then pose the research challenge to understand how we can build a knowledge graph starting from a data lake repository to improve its value (Q2). In this sense, the second contribution is precisely the generation of a Tourism Knowledge Graph (TKG) for Sardinia and London destinations. Indeed, the main role for TKG has been the provision of a semantic layer to an existing data lake, built within an industrial project about Tourism 4.0 called *Data Lake Turismo*, whose aim was collecting, transforming, and analysing data in this sector.

In order to support the knowledge graph generation three more contributions are provided in the present work. First, a novel ontology named Tourism Analytics Ontology (TAO) that was designed to provide a foundation for modelling all entities and relations in the Tourism Knowledge Graph. Then an open-source pipeline for generating this same knowledge graph from (semi-) structured and unstructured data stored in the data lake. Lastly, an evaluation procedure for assessing functional, logical, and structural dimensions of both TAO and the knowledge graph.

Continuing in our inquiry we considered language models as the more suitable form of artificial intelligence to create intelligent applications in the tourism domain and we tried to understand what methods are best suited to enhance Language Models with information from knowledge graphs in order to improve specific tasks like classification (Q3) and how can we leverage knowledge graphs together with Language Models to build intelligent applications (Q4) Following this objectives the thesis proposes three final contributions.

First, we introduce a comparative analysis of knowledge enhancement strategies for Language Models applied to the Scholarly Domain. This study offers a meticulous overview and a comparative evaluation of four predominant techniques, focusing on their effectiveness in the classification of scientific articles and paves the way for the use of similar approaches in the tourism domain.

Second, we present a novel methodology that effectively integrates language models and knowledge graphs in the context of the tourism domain, together with a comprehensive evaluation demonstrating the advantages of the proposed solution compared to conventional transformer models and an in-depth analysis of feature engineering techniques to identify the most effective combination of features when integrating knowledge graphs and language models.

In the end, we propose an example of an intelligent application that supports hospitality revenue managers in understanding the market positioning of their accommodation offerings and optimizing their proposition on online platforms.

Appendix A

Data Lake Turismo

All the data used in Chapter 3 are based on the *Data Lake Turismo* project. One of the main products of this project was a data lake created to collect, transform and analyze data about the tourism sector. The data lake has been built by Linkalab using Amazon Web Services (AWS) cloud computing.

Here we briefly describe the main characteristics of the data lake:

1. **data lake environment and data lake structure**: the data lake has been built using S3[1] cloud storage and other serverless AWS services[2]; it is structured in tiers (layers) and zones within each layer[3]; the data is always preserved in its raw format and transformed through data pipelines, which operate within or across tiers and zones, to be finally exposed in the consumption layer to external applications;

2. **data collection from the web**: the data lake platform collects data from many web sources via deep web crawling, wrapped-based extraction, and APIs; among others, Booking.com and AirBnB sources are harvested;

3. **data preparation**: raw data (HTML, JSON) collected in the intake zone of the data lake are transformed (cleaned, flattened, combined, enriched) to be ready to use for data analysis;

4. **data consumption**: the consumption of data can be performed either through direct access to files in the S3 cloud storage or through SQL queries to the Glue Data Catalog[4] service using Athena serverless query engine[5].

5. **data catalog**: the data lake has a technical data catalog based on AWS Glue Data Catalog service and traces data provenance implicitly with respect to the source of data (using specific data lake tiers, zones, and dataset locations) and with respect to time by partitioning data using dates.

[1] `https://aws.amazon.com/s3/`

[2] `https://aws.amazon.com/serverless/`

[3] In this context we are referring to tiers and zones in the data lake as portions of the cloud storage and the metadata store (data catalog).

[4] `https://aws.amazon.com/glue/`

[5] `https://aws.amazon.com/athena/`

Appendix B

Requirements and competence queries

This section elucidates the functional and non-functional requirements identified during the process of defining the domain ontology (TAO), as outlined in Section 3.4.3, and the pertinent use cases. Additionally, we delve into the Competency Questions and how they encapsulate functional prerequisites in a more hands-on manner. Lastly, we investigate the data available in the data lake that facilitated the compilation of the CQs.

B.1 Prerequisites

In order to be effectively utilized to construct a knowledge graph capable of facilitating the use cases identified in Section 3.4.1, we anticipated that the ontology would need to meet the following functional prerequisites (FR):

FR 1 model lodging facilities and establish a hierarchy[1] of their types (e.g., hotels, hostels, apartments),

FR 2 model accommodations and establish a hierarchy of their types (e.g., room, entire apartment, suite);

FR 3 model amenities provided to tourists and establish 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 establish a hierarchy of their types;

FR 5 model the associations among entities (e.g., geographic associations, mentions, composition/inclusion);

FR 6 model tourist reviews;

[1]For a description of hierarchies and their implementation in the TAO ontology see Appendix C.

Table B.1: Mapping knowledge graph's use cases with ontology's functional prerequisites

Use Case	FR1	FR2	FR3	FR4	FR5	FR6	FR7
UC1 - KG should facilitate the detection of the topics of interest discussed by tourists in their reviews	X	X	X	X		X	
UC2 - KG should facilitate the detection of the topics of interest illustrated in the descriptions of lodging facilities and accommodation offers	X	X	X	X			
UC3 - KG should facilitate the identification and linking of tourism entities in the KG for various applications revolving around the domain of social media, news, and blogs	X	X	X	X			
UC4 - KG should facilitate sentiment analysis applications about tourists' attitudes toward lodging businesses and destinations	X	X	X	X	X		X
UC5 - KG should facilitate the classification of tourist destinations based on their offerings and on tourist opinions	X	X	X	X	X	X	X

FR 7 model tourist destinations (e.g., Sardinia, London), which is the focal point of the trip.

The functional prerequisites for the ontology are linked to the use cases of the knowledge graph as outlined in Table B.1. For instance, we can observe that since *"KG should facilitate the detection of the topics of interest discussed by tourists in their reviews"* the ontology should design user reviews (FR6) and concepts typically related to what tourists talk about such as lodging facilities (FR1), accommodations (FR2), amenities (FR3), and tourist destinations (FR4).

In terms of non-functional prerequisites (NFR), the ontology should facilitate reasoning and be based on technical and market standards that are widely adopted. Specifically:

NFR 1 it should be defined in OWL[2];

[2]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.

NFR 2 it should be based on two *de-facto* standards for modeling business data:

- Schema.org[3], which is a set of vocabularies developed collaboratively for structuring data on the internet. It was originally established 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 it should be simple to extend in order to cater to other use cases in the tourism domain.

B.2 Competency Questions

We have defined the following 12 competency questions based on the functional requirements:

CQ 1 What are the first n (for instance, 10) accommodation facilities of a certain type (like hotels) that have more than m (for instance, 1,000) reviews and the lowest average value of user review scores?

CQ 2 Identify three apartments with a specific amenity (such as Wi-Fi), within a certain distance Km (say 2Km) from at least a certain number (say 2) of tourist attractions (like Parks).

CQ 3 Which Tourist Destinations have the highest percentage of expensive Lodging Facilities (providing at least one offer for two-person accommodation at a nightly price twice the average price)?

CQ 4 Which are the n (for instance, 10) tourist spots most frequently mentioned by hotel descriptions that also offer a specific amenity (like a day Spa) in a particular tourist destination?

CQ 5 Which Tourist Locations are most frequently mentioned in all Accommodation Facility descriptions within a given tourist destination?

CQ 6 Which are the Tourist Locations most frequently mentioned in positive user reviews?

CQ 7 What are the n (for instance, 10) least expensive apartments that offer at least m (for instance, 2) beds and a specific amenity (like secured parking) and are within a certain distance (like 10km) from a specific type of tourist attraction (like an airport)?

CQ 8 Which type of Accommodation Facility receives the most reviews from tourists in a given Tourist Destination?

[3]See `https://schema.org/`

Table B.2: Correlation between competency questions and 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 9 Which are the top Tourist Destinations in terms of positive sentiment regarding food (percentage of Accommodation Facilities with positive reviews mentioning food)?

CQ 10 During which months are there the most user reviews for accommodation facilities of a certain type (like hotels)?

CQ 11 Which Tourist Locations exist in a Tourist Destination?

CQ 12 What is the number of beds available for rent in a specific Tourist Destination?

As demonstrated, CQs can be either more general or specific, depending on which facet of the ontology we want to describe and eventually test. However, all CQs are formulated as questions that can be converted into SPARQL queries against the KG. That's why, in some CQs, we can use actual examples (like Wi-Fi) instead of more abstract entity classes (like "a location amenity").

Typically, a competency question includes information related to various functional requirements, and conversely, a specific functional requirement is addressed by several competency questions. The correlation between CQs and functional requirements is illustrated in Table B.2.

B.3 Data Source Information

The formulation of the competency questions was also influenced by the information available in the data sources. Below, we present a list of the most pertinent information available in the data sources (discussed in Section 3.4.2) that guided the formulation of the CQs:

1. Information about accommodation facilities:

   (a) Name(s)
   (b) Location
   (c) Geographic relationships with administrative divisions
   (d) Geographic relationships with tourist destinations
   (e) Category (e.g., Hotel, Resort, Motel, B&B, Holiday Accommodations)
   (f) Type of accommodation provided (e.g., room, apartment, villa, bungalow, etc.)
   (g) Amenities (e.g., sauna, parking, swimming pool, breakfast, air conditioning, etc.)
   (h) Prices for accommodation listed online
   (i) User ratings
   (j) Textual descriptions (for performing Named Entity Recognition, Entity Linking and Relation Extraction, etc.)

2. Information about tourist locations:
   (a) Name (in different languages)
   (b) Location
   (c) Geographic relationships with administrative divisions
   (d) Geographic relationships with tourist destinations

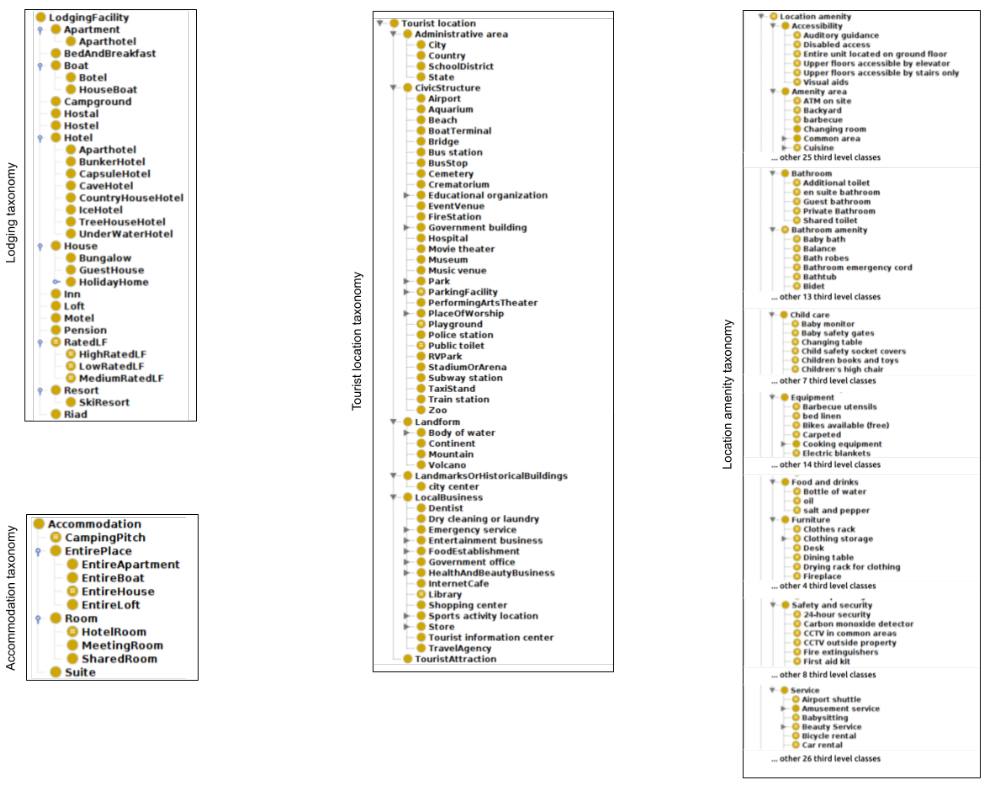
3. Information about tourist destinations:
   (a) Name (in different languages)
   (b) Location
   (c) Geographic relationships with administrative divisions
   (d) Geographic relationships with tourist locations

4. Tourist reviews about accommodation businesses and locations
   (a) User ratings
   (b) Tourist nationality and type of tourist (family, couple, etc.)
   (c) Textual review (for performing Named Entity Recognition, Entity Linking and Relation Extraction, etc.)

This list was used during the ontology engineering process, as it helps to define the entities and properties to be modeled by the TAO ontology.

Appendix C

Hierarchy Classes within TAO



Lodging taxonomy

- LodgingFacility
  - Apartment
    - Aparthotel
  - BedAndBreakfast
  - Boat
    - Botel
    - HouseBoat
  - Campground
  - Hostal
  - Hostel
  - Hotel
    - Aparthotel
    - BunkerHotel
    - CapsuleHotel
    - CaveHotel
    - CountryHouseHotel
    - IceHotel
    - TreeHouseHotel
    - UnderWaterHotel
  - House
    - Bungalow
    - GuestHouse
    - HolidayHome
  - Inn
  - Loft
  - Motel
  - Pension
  - RatedLF
    - HighRatedLF
    - LowRatedLF
    - MediumRatedLF
  - Resort
    - SkiResort
  - Riad

Accommodation taxonomy

- Accommodation
  - CampingPitch
  - EntirePlace
    - EntireApartment
    - EntireBoat
    - EntireHouse
    - EntireLoft
  - Room
    - HotelRoom
    - MeetingRoom
    - SharedRoom
  - Suite

Tourist location taxonomy

- Tourist location
  - Administrative area
    - City
    - Country
    - SchoolDistrict
    - State
  - CivicStructure
    - Airport
    - Aquarium
    - Beach
    - BoatTerminal
    - Bridge
    - Bus station
    - BusStop
    - Cemetery
    - Crematorium
    - Educational organization
    - EventVenue
    - FireStation
    - Government building
    - Hospital
    - Movie theater
    - Museum
    - Music venue
    - Park
    - ParkingFacility
    - PerformingArtsTheater
    - PlaceOfWorship
    - Playground
    - Police station
    - Public toilet
    - RVPark
    - StadiumOrArena
    - Subway station
    - TaxiStand
    - Train station
    - Zoo
  - Landform
    - Body of water
    - Continent
    - Mountain
    - Volcano
  - LandmarksOrHistoricalBuildings
    - city center
  - LocalBusiness
    - Dentist
    - Dry cleaning or laundry
    - Emergency service
    - Entertainment business
    - FoodEstablishment
    - Government office
    - HealthAndBeautyBusiness
    - InternetCafe
    - Library
    - Shopping center
    - Sports activity location
    - Store
    - Tourist information center
    - TravelAgency
  - TouristAttraction

Location amenity taxonomy

- Location amenity
  - Accessibility
    - Auditory guidance
    - Disabled access
    - Entire unit located on ground floor
    - Upper floors accessible by elevator
    - Upper floors accessible by stairs only
    - Visual aids
  - Amenity area
    - ATM on site
    - Backyard
    - barbecue
    - Changing room
    - Common area
    - Cuisine
  - ... other 25 third level classes
  - Bathroom
    - Additional toilet
    - en suite bathroom
    - Guest bathroom
    - Private Bathroom
    - Shared toilet
  - Bathroom amenity
    - Baby bath
    - Balance
    - Bath robes
    - Bathroom emergency cord
    - Bathtub
    - Bidet
  - ... other 13 third level classes
  - Child care
    - Baby monitor
    - Baby safety gates
    - Changing table
    - Child safety socket covers
    - Children books and toys
    - Children's high chair
  - ... other 7 third level classes
  - Equipment
    - Barbecue utensils
    - bed linen
    - Bikes available (free)
    - Carpeted
    - Cooking equipment
    - Electric blankets
  - ... other 14 third level classes
  - Food and drinks
    - Bottle of water
    - oil
    - salt and pepper
  - Furniture
    - Clothes rack
    - Clothing storage
    - Desk
    - Dining table
    - Drying rack for clothing
    - Fireplace
  - ... other 4 third level classes
  - Safety and security
    - 24-hour security
    - Carbon monoxide detector
    - CCTV in common areas
    - CCTV outside property
    - Fire extinguishers
    - First aid kit
  - ... other 8 third level classes
  - Service
    - Airport shuttle
    - Amusement service
    - Babysitting
    - Beauty Service
    - Bicycle rental
    - Car rental
  - ... other 26 third level classes

Figure C.1: **A graphical depiction of four hierarchies included within the TAO ontology, expanded to the third tier (some classes have been omitted in the location amenity hierarchy for the sake of clarity and space).**

TAO incorporates a variety of class hierarchies, which are linked through the property rdfs:subClassOf. We chose this method over alternatives (e.g., utilizing taxonomies via SKOS[1]) to ensure compatibility with the Accommodation Ontology (where types of accommodations and amenities are portrayed as subclasses) and to streamline the use of Schema.org, where class hierarchies are also employed. Specifically, we have constructed four hierarchies to detail the relationships between relevant classes, including:

1. the *lodging hierarchy*, which includes 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*, which includes 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`) spread over 5 levels;

Figure C.1 showcases the initial three levels of each hierarchy. For every subclass within a hierarchy, we can implement one or more of the following:

- if a class has a conceptual link to a similar class in other ontologies (e.g., DBpedia), we model this with the annotation property `rdfs:seeAlso`;
- if a class originates from other ontologies, we trace the source using the `dc:source` property to display the initial class[2];
- if a class extension[3] matches the extension of a class in other ontologies, we link them with the `owl:equivalentClass` property[4], or the `rdfs:subClassOf` property if it is more specific[5];
- for each class, we use `rdfs:label` to indicate the main label and `skos:altLabel` to show alternate labels;
- when appropriate, disjoint axioms are incorporated to improve reasoning.

C.1 Lodging taxonomy

The initial hierarchy outlines the various types of lodging facilities and their sub-types, such as Aparthotel, which is a sub-type of a hotel. We have also introduced a unique case with `tao:RatedLF` and its subclasses, which are used

[1]Refer to `https://www.w3.org/TR/2009/REC-skos-reference-20090818/`

[2]Bear in mind that `dc:` represents Dublin Core;

[3]The group of individuals that are members of the class.

[4]This applies to `tao:TouristDestination`, which is declared to be `owl:equivalentClass` of `schema:TouristDestination`

[5]This applies to `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.

to categorize *Lodging facilities* based on 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[6] on `tao:normRatingValue` for automatic classification of a *Lodging facility*.

A rated lodging facility is also part of `tao:RatedLF` (rated lodging facility) class[7] and it can be inferred whether it belongs to one of the following three sub-classes:

- part of `tao:HighRatedLF` class if it is associated[8] with a `tao:HighNormRating` node;
- part of `tao:MediumRatedLF` class if it is associated with a `tao:MediumNormRating` node;
- part of `tao:LowRatedLF` class if it is associated with a `tao:LowNormRating` node;

C.2 Accommodation hierarchy

In terms of modeling *accommodations*, we differentiated two general provisions: (i) the entire place (i.e., EntirePlace), and (ii) room (i.e., Room). We also established sub-classes for these (e.g., EntireHouse for EntirePlace, HotelRoom for Room). Furthermore, we modeled two unique cases (i.e., CampingPitch and Suite), which are not included in the general cases. When suitable, we employed equivalence axioms to add useful constraints, such as in the case of HotelRoom, which must be part of one Hotel. Moreover, to ensure high compatibility between TAO and the Accommodation Ontology, we defined the accommodation classes of TAO as subclasses of the ones in the Accommodation Ontology (e.g., `tao:CampingPitch` is a subclass of `acco:CampingPitch`).

C.3 Hierarchy of Location Amenities

For the *location amenities*, equivalence axioms are included to facilitate a level of mapping with the likely definitions of specific accommodation features using the Accommodation ontology approach[9]. Therefore, every subclass in this hierarchy is also declared as `owl:equivalentClass` to an anonymous class that aligns with the Accommodation Ontology prescriptions[10]. Thus, each anonymous class is defined as a subclass of `acco:AccommodationFeature` and as an `owl:intersectionOf` of `owl:Restriction` based

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

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

[8]Using `tao:aggregateNormRating` object property

[9]Due to the absence of a specific taxonomy and the use of a textual label to define a specific feature, it is more of an educated guesswork to identify the label most likely to be used.

[10]It is described as "a structured value representing the feature of an accommodation as a property-value pair of varying degrees of formality"; refer `http://ontologies.sti-innsbruck.at/acco/ns.html#AccommodationFeature`

on `gr:name` and `acco:value` data properties from GoodRelations. Below is an example 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 ;
       owl:hasValue "yes"@en
      ]
      [
       rdf:type owl:Restriction ;
       owl:onProperty gr:name ;
       owl:hasValue "Airport␣Shuttle"@en
      ]
    ) ;
  ] .
```

By this means, a reasoner can map to the appropriate `tao:LocationAmenity` sub-class an accommodation feature defined using `acco:value` and `gr:name` as per the Accommodation ontology guidelines.

C.4 Hierarchy of Tourist Locations

Tourist locations are conceptualized, wherever feasible, in accordance with their respective GeoNames feature codes. This is achieved by classifying them as `owl:equivalentClass` to an anonymous class that is a restriction on the property `gn:featureCode` that must possess a suitable value from the GeoName feature codes list[11]. The following is an example:

```
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 .
```

[11]Refer `https://www.geonames.org/export/codes.html`

Appendix D

Data Transformation

This Appendix details the procedures involved in modifying the data illustrated in Figure 3.4.

D.1 Data extraction

The initial step involved extracting pertinent data from the original data lake. The extraction was carried out using a big data SQL engine[1]. During this phase, the data is also amalgamated and organized for easier processing in subsequent stages (for example, unique identifiers are generated, and nested columns are expanded). This results in the creation of the *Source data* assets collection, which includes:

1. hospitality_supply_assets: this contains details about accommodation facilities and services.
2. hospitality_demand_assets: this contains user reviews.

D.2 Data breakdown and filter

This second stage structures and arranges the information generated in the previous stage. Specifically, we need to:

1. segment the information so that we have a unique asset for each semantic entity we want to represent as triples (e.g., accommodation facility, accommodation, service, review);
2. apply a flat structure to the data, as some columns include complex data structures such as arrays or key/value structures;
3. separate text blobs from the other data while maintaining their relation to the semantic entity they refer to (e.g., the accommodation facility description, the review content).

[1]Amazon Athena, see `https://aws.amazon.com/en/athena`

We can achieve the desired structure using dedicated data pipelines that generate multiple assets from a single one, flattening the data and filtering out unnecessary columns. This results in an unpacked version of the assets for each source:

1. hospitality_unpacked_supply_assets: containing unpacked details about accommodation facilities and services.
2. hospitality_unpacked_demand_assets: containing unpacked user reviews.

D.3 Data cleaning

In this step, we rectify or eliminate corrupt or inaccurate records from the assets generated in the preceding step. Specifically, we need to eliminate duplicate records, remove special characters, standardize categorical fields, normalize date and numeric fields.

From hospitality_unpacked_supply_assets, the Data Cleaning step yields:

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

Conversely, from hospitality_unpacked_demand_assets, the Data Cleaning generates:

1. reviews_assets - containing all structured data related to user reviews of an accommodation facility; a unique ID is generated for each review;
2. reviews_content_assets - containing all text content for user reviews of an accommodation facility (used for Named Entity Extraction and Linking);

D.4 Ontology mappings

In this phase, we discern and map the classes of the structured data to convert them into triples.

For example, if a lodging entity is represented as a record like:

Key	Value
hotel_id	9f40f613d308cf80
name	Chelsea BnB
structure_type	Bed and breakfast

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

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

Structured data incorporate categorical columns that refer to concepts in the TAO ontology. Specifically, there are three hierarchies in the ontology (Refer to Appendix C for further information) that we have to harmonize with categorical columns in the data:

1. accommodation facility types: for each accommodation table record we have a text field that holds the name of the accommodation facility type; this field can be used to associate the correct `tao:LodgingFacility` subclass to the individual accommodation facility the record pertains to;

2. accommodation types: for each accommodation table record we have a text field that holds 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 pertains to;

3. accommodation features (amenities) types: for each amenity table record we have an accommodation feature associated with a specific accommodation facility (via an external key ID that refers to the accommodation table). This field can be used to associate the correct `tao:LocationAmenity` subclass to the individual amenity the record pertains to.

To execute the reconciliation we employ a heuristic process based on rules that can identify the most suitable class to use to model an entity. The heuristic process utilizes 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 have already described in Section 3.4.3. The reconciliation is thus executed 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 phase produces new types of assets that form part of the *Ontology mapped data* asset collection:

1. classified_lodging_assets;

2. classified_accommodation_assets;

3. classified_amenities_assets.

These assets will be used in the triple creation process.

D.5 Language detection

In this phase, a language detection algorithm [224] is applied to the text contained in the accommodation 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 phases can process only English texts. The enriched assets form part of the *Language enriched data* asset collection.

D.6 Linking entities with DBpedia

In order to carry out the Entity Linking task with DBpedia, we utilized DBpedia Spotlight [218, 225] APIs[2] on the English text found in the tables containing lodging descriptions and reviews. DBpedia Spotlight recognizes and annotates entities through the following pipeline process:

- Spotting: Identifies potential entity mentions (surface forms) within the original input text.
- Candidate selection: Chooses DBpedia resources that could potentially be the meanings of each surface form.
- Disambiguation: Decides which candidate is the most likely resource for each surface form.
- Filtering: Customizes the annotation task based on the user's needs.

For the filtering stage, we limited the annotation scope to these entity types: `DBpedia:Activity, DBpedia:Food, DBpedia:Holiday, DBpedia:MeanOfTransportation, DBpedia:Place, Schema:Event, Schema:Place`. The DBpedia entity linking process resulted in two new kinds of assets that are part of the *DBpedia linked entities* asset collection:

1. lodging_dbpedia_linked_assets - Contains a record for each DBpedia entity linked to a lodging facility as defined by its unique ID;
2. review_dbpedia_linked_assets - Contains a record for each DBpedia entity linked to a user review as defined by its unique ID.

These assets were utilized in the process of creating triples.

D.7 Linking entities with GeoNames

This stage executes an Entity Linking task with GeoNames. It links places mentioned in lodging descriptions and reviews to their corresponding entities in GeoNames.

[2]`https://www.dbpedia.org/resources/spotlight/`

We used an open-source software called Mordecai[3] [226] for this purpose. Mordecai is a comprehensive geoparsing system that extracts place names from text, matches them to the correct entries in a gazetteer, and returns structured geographic information for the resolved place name. Mordecai uses a language-agnostic architecture and word2vec [46] to infer the correct country for a set of locations in a text. It uses a custom-built Elasticsearch database populated with GeoNames data as a gazetteer. Mordecai is integrated with the Spacy library[4]. As with DBpedia in Appendix D.6, we used Mordecai to process all English text in the lodging description and review content tables. The GeoNames entity linking process resulted in two new kinds of assets that are part of the *GeoNames linked entities* asset collection:

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

We used these assets in the process of creating triples.

D.8 Strategy for implementation

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

- Data-driven,
- Flexible and easily extendable,
- Scalable within a distributed computing environment,
- Simple to manage,
- Easily set up for lineage (also known as provenance) metadata collection.

In accordance with these requirements, the data computation is structured using the pipeline approach previously described. This method is ideal for creating a distributed computation if the intermediate and final materializations are stored on a distributed file system. This is the same approach used by Apache Spark and other big data frameworks.

To handle the execution of a series of data pipelines, we used Dagster[5], 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 began using a single machine to simplify the deployment process, without foregoing the option to switch to a distributed architecture later on. Dagster can also provide metadata about the execution of each pipeline and the created assets, enabling

[3]`https://github.com/openeventdata/mordecai`

[4]Currently, only Spacy v2.x is supported

[5]`https://dagster.io/`

our system to generate provenance information for the Knowledge Graph. The data transformation code is developed using the Python Pandas[6] library. We made the pipelines built on Dagster publicly available as an open-source resource[7].

D.9 Performance on a standalone server

We utilized a single node with an AMD Ryzen™ 7 5800H CPU, 32GB of RAM, a 1TB SSD, and Ubuntu 20.04. With this setup, the data transformation of the booking.com and Airbnb data took approximately 8 hours and 45 minutes. The entity linking process took 7 hours and 14 minutes, language detection took 1 hour and 26 minutes, and all other data extraction and transformation steps took only 14 minutes. This is because the entity linking is performed by calling DBpedia Spotlight public end-points, so we had to limit the concurrency of the requests to the external web service to avoid server-side errors. This is the primary limitation to scalability for the current 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, a self-managed setup for DBpedia Spotlight can be created as described on their website[8]. As for language detection, it can be optimized 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 disk space usage by 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 facilitate storage scalability, a distributed filesystem could be used as suggested in Appendix C.8.

[6]`https://pandas.pydata.org/`

[7]See `https://github.com/linkalab/tkg/tree/main/kg_pipelines`

[8]See `http://dev.dbpedia.org/Dbpedia_Spotlight`

Appendix E

In-depth examination of the triple structure for TKG

This Appendix offers a detailed look at how triples that represent lodging facilities, accommodations, offers, and user reviews are structured within the Tourism Knowledge Graph. We will make reference to Figure 3.5 in the subsequent sections.

E.1 Structural details of triples pertaining to lodging facility entities

Figure 3.5 allows us to focus our attention on the triples that model a lodging facility, which encompasses:

1. an address entity (`:address_1`), conceptualized as a `schema:PostalAddress` class, providing us with a versatile means of defining the facility's location;
2. one or several entities representing features of the accommodation that are linked to the lodging facility via the `tao:feature` property; in our instance, we have the node `:amenity_1` of the type `tao:Parking`[1].
3. an entity that represents aggregated ratings (`:agg_rating_1` in our case) that is employed to model the cumulative user rating for the lodging facility (which is tied to the ratings given by individual users' reviews) and denotes the score on a standardized scale from 0 to 1.

[1]Typically, the amenity's class should be the most suitable TAO ontology class out of all the subclasses of `tao:LocationAmenity`, as determined during the Ontology mapping phase explained in Appendix D.4

E.2 Structural details of triples pertaining to accommodation entities

Accommodation is invariably linked to a lodging facility, in accordance with the Accommodation ontology, and it comprises:

1. the maximum and minimum capacity for occupancy, using a `gr:QuantitativeValue` node (`:capacity_1` in our case);
2. the provision of beds, represented by a `acco:BedDatails` node (`:beds_1` in our case);
3. the specific type of accommodation[2] (utilizing one of the TAO ontology classes such as `tao:Room`).

E.3 Structural details of triples pertaining to offer entities

We detail a commercial offer for leasing an accommodation by taking advantage of GoodRelations. As depicted in Figure 3.5, an offer can be articulated in terms of:

1. a node (`:quantity_1`) of the type `gr:TypeAndQuantityNode` utilized to specify the duration of the offer in days using the `gr:amountOfThisGood` and `gr:hasUnitOfMeasurement` properties;
2. a node (`:price_spec_1`) of the type `gr:UnitPriceSpecification` used to detail the price and currency per day using the `gr:hasUnitOfMeasurement`, `gr:hasCurrency`, and `gr:hasCurrencyValue` properties.

E.4 Structural details of triples pertaining to user reviews entities

A user's review of the lodging facility is represented in TKG by a pair of entities:

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

[2]As identified during the Ontology mapping phase detailed in Appendix D.4

Appendix F

Expansion of TAO

The Python code example we present here demonstrates how we have utilized owlready2 to broaden the scope of the TAO ontology by introducing new classes:

Listing F.1: Python code to expand the TAO ontology with new categories.

```
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 subsequent code section, we provide an illustration of the processing of a CSV file that outlines new classes for integration into the ontology. All information from the CSV file is loaded into a pandas dataframe and manipulated by a bespoke function (process_entity function). This function employs owlready2 for the management of OWL class creation or alteration. For a comprehensive view, refer to the complete source code.

Listing F.2: Python code to expand the TAO ontology with new categories.

```
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")
```

Bibliography

[1] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo Salatino, and Luca Secchi. Enriching Data Lakes with Knowledge Graphs. *CEUR Workshop Proceedings*, 3184:123–131, 2022.

[2] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo Salatino, and Luca Secchi. Data-driven methodology for knowledge graph generation within the tourism domain. *IEEE Access*, 11:67567–67599, 2023.

[3] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. Optimizing tourism accommodation offers by integrating language models and knowledge graph technologies. 2023.

[4] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. Language models enhancement with knowledge graphs for classification tasks within the tourism domain. 2023.

[5] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. A comparative analysis of knowledge injection strategies for large language models in the scholarly domain. 2023.

[6] Andrea Cadeddu, Alessandro Chessa, Vincenzo De Leo, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. Language models enhancement with knowledge graphs through knowledge injection. 2023.

[7] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[8] Richard Cyganiak, David Hyland-Wood, and Markus Lanthaler. Rdf 1.1 concepts and abstract syntax. *W3C Proposed Recommendation*, 01 2014.

[9] Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, Lynn Andrea Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10(2):1–53, 2004.

[10] SPARQL 1.1 Query Language. Technical report, W3C, 2013.

[11] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Knowledge Graphs. *Knowledge Graphs*, mar 2020.

[12] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *16th International World Wide Web Conference, WWW2007*, pages 697–706, 2007.

[13] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing wikidata to the linked data web. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, pages 50–65, Cham, 2014. Springer International Publishing.

[14] James Dixon. Pentaho, hadoop, and data lakes. *blogpost*, 2010.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(September 2016):1–4, 2016.

[18] Piero A. Bonatti, S. Decker, Axel Polleres, and Valentina Presutti. Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371). *Dagstuhl Reports*, 8:29–111, 2019.

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

[20] Kurt D. Bollacker, Patrick Tufts, Tom Pierce, and Robert Cook. A platform for scalable, collaborative, structured information integration. 2007.

[21] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, sep 2014.

[22] Amit Singhal. Introducing the knowledge graph: things, not strings. *Google blog*, 2012.

[23] Arun Krishnan. Making search easier: How amazon's product graph is helping customers find products more easily. *Amazon Blog*, 2018.

[24] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5), sep 2017.

[25] Martin J. Dürst and Michel Suignard. Internationalized resource identifiers (iris). *RFC*, 3987:1–46, 2005.

[26] Piero A. Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Journal of Web Semantics*, 9(2):165–201, 2011. Provenance in the Semantic Web.

[27] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. *Int. J. Semantic Web Inf. Syst.*, 5:1–24, 04 2009.

[28] Tim Berners-Lee. Linked data - design issues. *W3C*, (09/20), 2006.

[29] Dan Brickley, Ramanathan V Guha, and Andrew Layman. Resource description framework (rdf) schema specification. Technical report, Technical report, W3C, 1999. W3C Proposed Recommendation. http://www. w3 . . . , 1998.

[30] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. Shape expressions: An rdf validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, page 32–40, New York, NY, USA, 2014. Association for Computing Machinery.

[31] Holger Knublauch and Dimitris Kontokostas. Shapes constraint language (shacl). Technical report, Technical report, W3C, 2017. W3C Proposed Recommendation. https://www.w3.org/TR/2017/REC-shacl-20170720/, 2017.

[32] Frank Manola, Eric Miller, Brian McBride, et al. Rdf primer. *W3C recommendation*, 10(1-107):6, 2004.

[33] Natalia Miloslavskaya and Alexander Tolstoy. Big data, fast data and data lake concepts. *Procedia Computer Science*, 88:300–305, 2016. 7th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2016, held July 16 to July 19, 2016 in New York City, NY, USA.

[34] Corinna Giebler, Christoph Gröger, Eva Hoos, Holger Schwarz, and Bernhard Mitschang. Leveraging the Data Lake - Current State and Challenges. In *Proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2019)*, 2019.

[35] James Dixon. Data lakes revisited. *blogpost*, 2010.

[36] Huang Fang. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 820–824, 2015.

[37] Cedrine Madera and Anne Laurent. The next information architecture evolution: The data lake wave. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, MEDES, page 174–180, New York, NY, USA, 2016. Association for Computing Machinery.

[38] Daniel E. O'Leary. Embedding ai and crowdsourcing in the big data lake. *IEEE Intelligent Systems*, 29(5):70–73, 2014.

[39] Ignacio G. Terrizzano, Peter M. Schwarz, Mary Roth, and John E. Colino. Data wrangling: The challenging yourney from the wild to the lake. In *Conference on Innovative Data Systems Research*, 2015.

[40] Christian Mathis. Data lakes. *Datenbank-Spektrum*, 17:1–5, 10 2017.

[41] B. Sharma and an O'Reilly Media Company Safari. *Architecting Data Lakes, 2nd Edition*. O'Reilly Media, Incorporated, 2018.

[42] A. Gorelik. *The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science*. O'Reilly Media, 2019.

[43] Bill Inmon. *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*. Technics Publications, LLC, Denville, NJ, USA, 1st edition, 2016.

[44] Christian Mathis. Data lakes. *Datenbank-Spektrum*, 17(3):289–293, 2017.

[45] Mandy Chessell, Ferd Scheepers, Nhan Nguyen, Ruud van Kessel, and Ron van der Starre. Governing and managing big data for analytics and decision makers. *IBM Redguides for Business Leaders*, 252, 2014.

[46] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[47] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[48] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[49] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2873–2879. AAAI Press, 2016.

[50] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING*, pages 3485–3495. ACL, 2016.

[51] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[52] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63:1872–1897, 10 2020.

[53] Qi Liu, Matt J. Kusner, and Phil Blunsom. A survey on contextual embeddings. *ArXiv*, abs/2003.07278, 2020.

[54] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. Ammus : A survey of transformer-based pretrained models in natural language processing, 2021.

[55] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[57] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[58] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[59] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, 2019.

[60] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[61] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s), sep 2022.

[62] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao. A survey on vision transformer. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(01):87–110, jan 2023.

[63] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. Wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[64] Aswin Sivaraman and Minje Kim. Self-supervised learning from contrastive mixtures for personalized speech enhancement. *arXiv preprint arXiv:2011.03426*, 2020.

[65] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[66] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[67] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[68] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[69] Alexis Conneau and Guillaume Lample. *Cross-Lingual Language Model Pretraining.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[70] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[71] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[72] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[73] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics, 2018.

[74] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net, 2020.

[75] Luca Di Liello, Matteo Gabburo, and Alessandro Moschitti. Effective pretraining objectives for transformer-based autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5533–5547, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[76] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Knowledge inheritance for pre-trained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3921–3937, Seattle, United States, July 2022. Association for Computational Linguistics.

[77] Zhengyan Zhang, Yuxian Gu, Xu Han, Shengqi Chen, Chaojun Xiao, Zhenbo Sun, Yuan Yao, Fanchao Qi, Jian Guan, Pei Ke, Yanzheng Cai, Guoyang Zeng, Zhixing Tan, Zhiyuan Liu, Minlie Huang, Wentao Han, Yang Liu, Xiaoyan Zhu, and Maosong Sun. Cpm-2: Large-scale cost-effective pre-trained language models. *AI Open*, 2:216–224, 2021.

[78] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[79] Yuri Kuratov and Mikhail Arkhipov. Adaptation of deep bidirectional multilingual transformers for russian language. *ArXiv*, abs/1905.07213, 2019.

[80] Diedre Carmo, Marcos Piau, Israel Campiotti, Rodrigo Nogueira, and Roberto Lotufo. PTT5: Pretraining and validating the T5 model on Brazilian Portuguese data. *arXiv e-prints*, page arXiv:2008.09144, August 2020.

[81] Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains, 2021.

[82] Yichu Zhou and Vivek Srikumar. A closer look at how fine-tuning changes BERT. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[83] Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online, November 2020. Association for Computational Linguistics.

[84] Marius Mosbach, Anna Khokhlova, Michael A. Hedderich, and Dietrich Klakow. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 68–82, Online, November 2020. Association for Computational Linguistics.

[85] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Investigating learning dynamics of BERT fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China, December 2020. Association for Computational Linguistics.

[86] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to Fine-Tune BERT for Text Classification? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11856 LNAI(2):194–206, 2019.

[87] Santiago González-Carvajal and Eduardo Garrido-Merchán. Comparing bert against traditional machine learning text classification. *Journal of Computational and Cognitive Engineering*, 05 2020.

[88] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46-47:2–13, 2017.

[89] Davide Gazzè, Angelica Lo Duca, Andrea Marchetti, and Maurizio 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.

[90] Pablo Calleja, Freddy Priyatna, Nandana Mihindukulasooriya, and Mariano Rico. DBtravel: A tourism-oriented semantic graph. In Cesare Pautasso, Fernando Sánchez-Figueroa, Kari Systä, and Juan Manuel 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.

[91] Maritzol Tenemaza, José Limaico, and Sergio Luján-Mora. Tourism recommender system based on natural language classifier. In Tareq Z. Ahram, Waldemar Karwowski, and Jay Kalra, editors, *Advances in Artificial Intelligence, Software and Systems Engineering*, pages 230–235, Cham, 2021. Springer International Publishing.

[92] Roopesh L R and Tulasi Bomatpalli. A survey of travel recommender system. *International Journal of Computer Sciences and Engineering*, 7(3):356–362, 2019.

[93] Jiaying Lyu, Asif Khan, Sughra Bibi, Jin Hooi Chan, and Xiaoguang Qi. Big data in action: An overview of big data studies in tourism and hospitality literature. *Journal of Hospitality and Tourism Management*, 51:346–360, 2022.

[94] Ada Bagozi, Devis Bianchini, Valeria De Antonellis, Massimiliano Garda, and Michele Melchiori. Personalised exploration graphs onÂ semantic data lakes. In Hervé Panetto, Christophe Debruyne, Martin Hepp, Dave Lewis, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems: OTM 2019 Conferences*, pages 22–39, Cham, 2019. Springer International Publishing.

[95] Claudia Diamantini, Domenico Potena, and Emanuele Storti. A semantic data lake model for analytic query-driven discovery. In *The 23rd International Conference on Information Integration and Web Intelligence*, iiWAS2021, page 183–186, New York, NY, USA, 2021. Association for Computing Machinery.

[96] Jasim Waheed Ansari. Semantic profiling in data lake. *RWTH Aachen University, Germany*, 81, 2018.

[97] Henrik Dibowski, Stefan Schmid, Yulia Svetashova, Cory Henson, and Tuan Tran. Using semantic technologies to manage a data lake: Data catalog, provenance and access control. In *Proceedings of the 13th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2020)*, 11 2020.

[98] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van De Walle. RML: A generic language for integrated RDF mappings of heterogeneous data. In *CEUR Workshop Proceedings*, volume 1184, 2014.

[99] Anastasia Dimou, Tom De Nies, Ruben Verborgh, Erik Mannens, and Rik 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.

[100] Mohamed Nadjib Mami. *Strategies for a Semantified Uniform Access to Large and Heterogeneous Data Sources.* PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, February 2021.

[101] AndrÃ© Pomp, Alexander Paulus, Andreas Kirmse, Vadim Kraus, and Tobias Meisen. Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures. *Technologies*, 6(3), 2018.

[102] Weizhen Zhang, Han Cao, Fei Hao, Lu Yang, Muhib Ahmad, and Yifei Li. The Chinese Knowledge Graph on Domain-Tourism. *Lecture Notes in Electrical Engineering*, 590(November):20–27, 2020.

[103] Ricardo Alonso-Maturana, Elena Alvarado-Cortes, Susana López-Sola, María Ortega Martínez-Losa, and Pablo 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.

[104] Sergio Consoli, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Silvio Peroni, Misael Mongiovì, and Aldo Gangemi. Producing linked data for smart cities: The case of catania. *Big Data Res.*, 7:1–15, 2017.

[105] Marcírio Silveira Chaves and Cássia Trojahn. Towards a multilingual ontology for ontology-driven content mining in Social Web sites. *CEUR Workshop Proceedings*, 687, 2010.

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

[107] Juan F. Sequeda, Willard J. Briggs, Daniel P. Miranker, and Wayne P. Heideman. A pay-as-you-go methodology to design and build enterprise knowledge graphs from relational databases. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, pages 526–545, Cham, 2019. Springer International Publishing.

[108] Gytundefined Tamašauskaitė and Paul Groth. Defining a knowledge graph development process through a systematic review. *ACM Trans. Softw. Eng. Methodol.*, feb 2022. Just Accepted.

[109] Danilo Dessì, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, and Enrico 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.

[110] Danilo Dessì, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, Enrico Motta, and Harald Sack. AI-KG: an automatically generated knowledge graph of artificial intelligence. In Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana 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.

[111] Elias Kärle, Umutcan Şimşek, Oleksandra Panasiuk, and Dieter Fensel. Building an ecosystem for the tyrolean tourism knowledge graph. *arXiv*, pages 260–267, 2018.

[112] Dinghe Xiao, Nannan Wang, Jiangang Yu, Chunhong Zhang, and Jiaqi 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.

[113] Mattia Atzeni and Diego Reforgiato Recupero. Multi-domain sentiment analysis with mimicked and polarized word embeddings for human-robot interaction. *Future Gener. Comput. Syst.*, 110:984–999, 2020.

[114] Amna Dridi and Diego Reforgiato Recupero. Leveraging semantics for sentiment polarity detection in social media. *Int. J. Mach. Learn. Cybern.*, 10(8):2045–2055, 2019.

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

[116] M Grüninger, Mark S Fox, and Michael 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.

[117] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*, page 25, 2001.

[118] Oliver Fodor and Hannes Werthner. Harmonise: A step toward an interoperable e-tourism marketplace. *International Journal of Electronic Commerce*, 9(2):11–39, 2005.

[119] Shiyan Ou, Viktor Pekar, Constantin Orasan, Christian Spurk, and Matteo 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.

[120] Steffen Staab, Christian Braun, Ilvio Bruder, Antje Düsterhöft, Andreas Heuer, Meike Klettke, Günter. Neumann, Bernd Prager, Jan Pretzel, Hans Peter Schnurr, Rudi Studer, Hans Uszkoreit, and Burkhard 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.

[121] Robert Barta, Christina Feilmayr, Birgit Pröll, Christoph Grün, and Hannes Werthner. Covering the semantic space of tourism : An approach based on modularized ontologies. *ACM International Conference Proceeding Series*, 2009.

[122] R. V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: Evolution of Structured Data on the Web. *Communications of the ACM*, 59(2):44–51, jan 2016.

[123] Martin 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.

[124] Elias Kärle, Umutcan Simsek, Zaenal Akbar, Martin Hepp, and Dieter Fensel. Extending the schema.org vocabulary for more expressive accommodation annotations. In Roland Schegg and Brigitte Stangl, editors, *Information and Communication Technologies in Tourism 2017*, pages 31–41, Cham, 2017. Springer International Publishing.

[125] Marcirio Silveira Chaves, Larissa Freitas, and Renata 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.

[126] Jean Baptiste 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.

[127] Luigi Iannone, Alan Rector, and Robert Stevens. Embedding Knowledge Patterns into OWL. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, editors, *The Semantic Web: Research and Applications*, pages 218–232, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[128] Martin G. Skjæveland, Henrik Forssell, Johan W. Klüwer, Daniel P. Lupp, Evgenij Thorstensen, and Arild Waaler. Pattern-based ontology design and instantiation with reasonable ontology templates. In *WOP@ISWC*, 2017.

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

[130] Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben 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.

[131] Ruben Verborgh, Miel Vander Sande, Pieter Colpaert, Sam Coppens, Erik Mannens, and Rik Van De Walle. Web-scale querying through linked data fragments. In *CEUR Workshop Proceedings*, volume 1184, 2014.

[132] Ruben Verborgh, Olaf Hartig, Ben De Meester, Gerald Haesendonck, Laurens De Vocht, Miel Vander Sande, Richard Cyganiak, Pieter Colpaert, Erik Mannens, and Rik Van de Walle. Querying datasets on the web with high availability. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, pages 180–196, Cham, 2014. Springer International Publishing.

[133] Aldo Gangemi, Carola Catenacci, Massimilano Ciaramita, and Jos Lehmann. Modelling Ontology Evaluation and Valilidation - in Proceedings of ESWC2006. *Eswc2006*, page 15, 2006.

[134] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Andrea Giovanni Nuzzolese, Valentina Presutti, and Chiara Veninata. Pattern-based design applied to cultural heritage knowledge graphs. *Semantic Web*, 12(2):313–357, 2021.

[135] Eva Blomqvist, Azam Seil Sepour, and Valentina 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.

[136] A.M. Orme, H. Tao, and L.H. Etzkorn. Coupling metrics for ontology-based system. *IEEE Software*, 23(2):102–108, mar 2006.

[137] Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat Pérez, Thomas Manhardt, Hassan Chafi, Mihai Capotă, Narayanan Sundaram, Michael Anderson, Ilie Gabriel Tănase, Yinglong Xia, Lifeng Nai, and Peter Boncz. LDBC graphalytics: A benchmark for large scale graph analysis on parallel and distributed platforms. *Proceedings of the VLDB Endowment*, 9(13):1317–1328, 2015.

[138] Alfredo Cuzzocrea and Il Yeol Song. Big graph analytics: The state of the art and future research agenda. *DOLAP 2014 - Proceedings of the ACM 17th International Workshop on Data Warehousing and OLAP, co-located with CIKM 2014*, pages 99–101, 2014.

[139] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–26, 2021.

[140] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana 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.

[141] Maximilian Nickel, Volker Tresp, and Hans-Peter 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.

[142] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume 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.

[143] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[144] Agustín Borrego, Daniel Ayala, Inma Hernández, Carlos R. Rivero, and David Ruiz. Cafe: Knowledge graph completion using neighborhood-aware features. *Engineering Applications of Artificial Intelligence*, 103:104302, 2021.

[145] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. Few-shot knowledge graph completion. *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

[146] OpenAI. Gpt-4 technical report, 2023.

[147] Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. Performance of chatgpt on usmle: Potential for ai-assisted medical education using large language models. *PLoS digital health*, 2(2):e0000198, 2023.

[148] Shang Gao, Mohammed Alawad, M Todd Young, John Gounley, Noah Schaefferkoetter, Hong Jun Yoon, Xiao-Cheng Wu, Eric B Durbin, Jennifer Doherty, Antoinette Stroup, et al. Limitations of transformers on clinical text classification. *IEEE journal of biomedical and health informatics*, 25(9):3596–3607, 2021.

[149] Krishna Kumar. Geotechnical parrot tales (gpt): Overcoming gpt hallucinations with prompt engineering for geotechnical applications. *arXiv preprint arXiv:2304.02138*, 2023.

[150] Sang-Woon Kim and Joon-Min Gil. Research paper classification systems based on tf-idf and lda schemes. *Human-centric Computing and Information Sciences*, 9:1–21, 2019.

[151] Angelo A. Salatino, Francesco Osborne, Aliaksandr Birukou, and Enrico Motta. Improving editorial workflow and metadata quality at springer nature. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, pages 507–525, Cham, 2019. Springer International Publishing.

[152] Hussam Alkaissi and Samy I McFarlane. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2), 2023.

[153] Tommaso Caselli, Valerio Basile, Jelena Mitrovic, and Michael Granitzer. Hatebert: Retraining BERT for abusive language detection in english. *CoRR*, abs/2010.12472, 2020.

[154] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics.

[155] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746, 2019.

[156] Spyretta Leivaditi, Julien Rossi, and Evangelos Kanoulas. A benchmark for lease contract review. *CoRR*, abs/2010.10386, 2020.

[157] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. AMMUS : A survey of transformer-based pretrained models in natural language processing. *CoRR*, abs/2108.05542, 2021.

[158] Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. A survey of knowledge enhanced pre-trained models. *CoRR*, abs/2110.00269, 2021.

[159] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. Knowledge graphs: opportunities and challenges. *Artificial Intelligence Review*, pages 1–32, 2023.

[160] Danilo Dessí, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, and Enrico Motta. Cs-kg: A large-scale knowledge graph of research entities and claims in computer science. In *The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*, pages 678–696. Springer, 2022.

[161] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo A. Salatino, and Luca Secchi. Data-driven methodology for knowledge graph generation within the tourism domain. *IEEE Access*, 11:67567–67599, 2023.

[162] Simone Angioni, Angelo Salatino, Francesco Osborne, Diego Reforgiato Recupero, and Enrico Motta. Aida: A knowledge graph about research dynamics in academia and industry. *Quantitative Science Studies*, pages 1–43, 2021.

[163] Angelo A. Salatino, Francesco Osborne, and Enrico Motta. Augur: Forecasting the emergence of new research topics. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, JCDL '18, page 303–312, New York, NY, USA, 2018. Association for Computing Machinery.

[164] F. Löffler, V. Wesp, S. Babalou, P. Kahn, R. Lachmann, B. Sateli, R. Witte, and B. König-Ries. Scholarlensviz: A visualization framework for transparency in semantic user profiles. In Kerry Taylor, Rafael Gonçalves, Freddy Lecue, and Jun Yan, editors, *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC).*, 2020.

[165] Xiaoyu Zhang, Senthil Chandrasegaran, and Kwan-Liu Ma. Conceptscope: Organizing and visualizing knowledge in documents based on domain ontology. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2021.

[166] Thanasis Vergoulis, Serafeim Chatzopoulos, Theodore Dalamagas, and Christos Tryfonopoulos. Veto: Expert set expansion in academia. In Mark Hall, Tanja Merčun, Thomas Risse, and Fabien Duchateau, editors, *Digital Libraries for Open Knowledge*, pages 48–61, Cham, 2020. Springer International Publishing.

[167] Marc Beck, Syed Tahseen Raza Rizvi, Andreas Dengel, and Sheraz Ahmed. From automatic keyword detection to ontology-based topic modeling. In *International Workshop on Document Analysis Systems*, pages 451–465. Springer, 2020.

[168] Serafeim Chatzopoulos, Thanasis Vergoulis, Ilias Kanellos, Theodore Dalamagas, and Christos Tryfonopoulos. Artsim: improved estimation of current impact for recent articles. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pages 323–334. Springer, 2020.

[169] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. *CoRR*, abs/1909.07606, 2019.

[170] Malte Ostendorff, Peter Bourgonje, Maria Berger, Julián Moreno Schneider, Georg Rehm, and Bela Gipp. Enriching BERT with knowledge graph embeddings for document classification. *CoRR*, abs/1909.08402, 2019.

[171] Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. SentiLARE: Sentiment-aware language representation learning with linguistic knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988, Online, November 2020. Association for Computational Linguistics.

[172] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[173] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online, November 2020. Association for Computational Linguistics.

[174] Yusheng Su, Xu Han, Zhengyan Zhang, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. Cokebert: Contextual knowledge selection and embedding towards enhanced pre-trained language models. *AI Open*, 2:127–134, 2021.

[175] Vivek Kumar, Diego Reforgiato Recupero, Rim Helaoui, and Daniele Riboni. K-LM: knowledge augmenting in language models within the scholarly domain. *IEEE Access*, 10:91802–91815, 2022.

[176] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3660–3670, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[177] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy, July 2019. Association for Computational Linguistics.

[178] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[179] Yujia Qin, Yankai Lin, Ryuichi Takanobu, Zhiyuan Liu, Peng Li, Heng Ji, Minlie Huang, Maosong Sun, and Jie Zhou. ERICA: Improving entity and relation understanding for pre-trained language models via contrastive learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume*

1: Long Papers), pages 3350–3363, Online, August 2021. Association for Computational Linguistics.

[180] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[181] Mandar Joshi, Kenton Lee, Yi Luan, and Kristina Toutanova. Contextualized representations using textual encyclopedic knowledge, 2021.

[182] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192, 2022.

[183] Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. Rulebert: Teaching soft rules to pre-trained language models. *ArXiv*, abs/2109.13006, 2021.

[184] Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. Skill: structured knowledge infusion for large language models. *arXiv preprint arXiv:2205.08184*, 2022.

[185] Angelo Salatino, Francesco Osborne, and Enrico Motta. Cso classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics. *International Journal on Digital Libraries*, pages 1–20, 2022.

[186] Francesco Osborne and Enrico Motta. Klink-2: Integrating multiple web sources to generate semantic topic networks. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015*, pages 408–424, Cham, 2015. Springer International Publishing.

[187] Francesco Osborne, Enrico Motta, and Paul Mulholland. Exploring scholarly data with rexplore. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, pages 460–477, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[188] Antonello Meloni, Simone Angioni, Angelo Salatino, Francesco Osborne, Diego Reforgiato Recupero, and Enrico Motta. Integrating conversational agents and knowledge graphs within the scholarly domain. *IEEE Access*, 11:22468–22489, 2023.

[189] Syed Tahseen Raza Rizvi, Sheraz Ahmed, and Andreas Dengel. Ace 2.0: A comprehensive tool for automatic extraction, analysis, and digital profiling of the researchers in scientific communities. *Social Network Analysis and Mining*, 13(1):81, 2023.

[190] Mardiah Mardiah, Shelvie Nidya Neyman, et al. Aggregate functions in categorical data skyline search (cdss) for multi-keyword document search. *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika*, 9(1), 2023.

[191] Tanay Aggarwal, Angelo Salatino, Francesco Osborne, and Enrico Motta. R-classify: Extracting research papers' relevant concepts from a controlled vocabulary. *Software Impacts*, 14, December 2022. Publisher: Elsevier.

[192] Jorge Chamorro-Padial and Rosa Rodríguez-Sánchez. Attention-survival score: A metric to choose better keywords and improve visibility of information. *Algorithms*, 16(4), 2023.

[193] Thiviyan Thanapalasingam, Francesco Osborne, Aliaksandr Birukou, and Enrico Motta. Ontology-based recommendation of editorial products. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018*, pages 341–358, Cham, 2018. Springer International Publishing.

[194] Marcos Vinícius Macêdo Borges and Julio Cesar dos Reis. Semantic-enhanced recommendation of video lectures. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, volume 2161, pages 42–46. IEEE, 2019.

[195] Kanyao Han. Incorporating knowledge resources into natural language processing techniques to advance academic research and application development. *University of Illinois at Urbana-Champaign*, 2023.

[196] Shruthi Chari, Oshani Seneviratne, Mohamed Ghalwash, Sola Shirai, Daniel M. Gruen, Pablo Meyer, Prithwish Chakraborty, and Deborah L. McGuinness. Explanation Ontology: A general-purpose, semantic representation for supporting user-centered explanations. *Semantic Web*, Preprint(Preprint):1–31, 2023. Publisher: IOS Press.

[197] Anderson Rossanez, Júlio Cesar dos Reis, and Ricardo da Silva Torres. Representing scientific literature evolution via temporal knowledge graphs. In *MEPDaW@ISWC*, 2020.

[198] Wentao Li, Huachi Zhou, Junnan Dong, Qinggang Zhang, Qing Li, George Baciu, Jiannong Cao, and Xiao Huang. Constructing low-redundant and high-accuracy knowledge graphs for education. In *Learning Technologies and Systems: 21st International Conference on Web-Based Learning, ICWL 2022, and 7th International Symposium on Emerging Technologies for Education, SETE 2022, Tenerife, Spain, November 21–23, 2022, Revised Selected Papers*, page 148–160, Berlin, Heidelberg, 2023. Springer-Verlag.

[199] Mojtaba Nayyeri, Gokce Muge Cil, Sahar Vahdati, Francesco Osborne, Mahfuzur Rahman, Simone Angioni, Angelo Salatino, Diego Reforgiato Recupero, Nadezhda Vassilyeva, Enrico Motta, et al. Trans4e: Link prediction on scholarly knowledge graphs. *Neurocomputing*, 461:530–542, 2021.

[200] Angelo Salatino, Simone Angioni, Francesco Osborne, Diego Reforgiato Recupero, and Enrico Motta. Diversity of expertise is key to scientific impact: a large-scale analysis in the field of computer science. In *Proceedings of the 27th International Conference on Science, Technology and Innovation Indicators (STI 2023)*, 2023.

[201] Simone Angioni, Angelo Salatino, Francesco Osborne, Diego Reforgiato Recupero, and Enrico Motta. The aida dashboard: Analysing conferences with semantic technologies. In *19th International Semantic Web Conference (ISWC 2020)*, 2020.

[202] Angelo A. Salatino, Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. The computer science ontology: A large-scale taxonomy of research areas. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018*, pages 187–205, Cham, 2018. Springer International Publishing.

[203] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L Opdahl, and Csaba Veres. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881, 2020.

[204] Angelo A. Salatino, Francesco Osborne, Thiviyan Thanapalasingam, and Enrico Motta. The cso classifier: Ontology-driven detection of research topics in scholarly articles. In Antoine Doucet, Antoine Isaac, Koraljka Golub, Trond Aalberg, and Adam Jatowt, editors, *Digital Libraries for Open Knowledge*, pages 296–311, Cham, 2019. Springer International Publishing.

[205] Pengfei Liu, Graham Neubig, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train , Prompt , and Predict : A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.*, 55:1–46, 2023.

[206] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[207] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305, 2020.

[208] Lydia González-Serrano and Pilar Talón-Ballestero. *Revenue Management and E-Tourism: The Past, Present and Future*, pages 1–28. Springer International Publishing, Cham, 2020.

[209] Marcello M. Mariani, Novin Hashemi, and Jochen Wirtz. Artificial intelligence empowered conversational agents: A systematic literature review and research agenda. *Journal of Business Research*, 161:113838, 2023.

[210] Pascal Hitzler. A review of the semantic web field. *Communications of the ACM*, 64(2):76–83, 2021.

[211] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

[212] Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. A survey of knowledge enhanced pre-trained models. *CoRR*, abs/2110.00269, 2021.

[213] Yan Xu, Mahdi Namazifar, Devamanyu Hazarika, Aishwarya Padmakumar, Yang Liu, and Dilek Hakkani-Tür. Kilm: Knowledge injection into encoder-decoder language models, 2023.

[214] Denis Emelin, Daniele Bonadiman, Sawsan Alqahtani, Yi Zhang, and Saab Mansour. Injecting domain knowledge in language models for task-oriented dialogue systems, 2022.

[215] Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. SKILL: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States, July 2022. Association for Computational Linguistics.

[216] Liang (Rebecca) Tang, Jaewook Kim, and Xi Wang. Estimating spatial effects on peer-to-peer accommodation prices: Towards an innovative hedonic model approach. *International Journal of Hospitality Management*, 81:43–53, 2019.

[217] Mochammad Agus Afrianto and Meditya Wasesa. Booking prediction models for peer-to-peer accommodation listings using logistics regression, decision tree, k-nearest neighbor, and random forest classifiers. *JISEBI*, 6(2):123–32, 2020.

[218] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian 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.

[219] Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021.

[220] Arian Askari, Amin Abolghasemi, Gabriella Pasi, Wessel Kraaij, and Suzan Verberne. Injecting the bm25 score as text improves bert-based re-rankers. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part I*, page 66–83, Berlin, Heidelberg, 2023. Springer-Verlag.

[221] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[222] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *CoRR*, abs/2002.0, 2020.

[223] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”why should i trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[224] Nakatani Shuyo. Language detection library for java, 2010.

[225] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. *ACM International Conference Proceeding Series*, pages 121–124, 2013.

[226] Andrew Halterman. Mordecai: Full Text Geoparsing and Event Geocoding. *The Journal of Open Source Software*, 2(9):91, 2017.