Universidade Estadual de Campinas
Instituto de Computação

Jesamin Melissa Zevallos Quispe

UpKG: A Framework to Insert New Domains in Knowledge Graphs

UpKG: Um Framework para Inserir Novos Domínios em Grafos de Conhecimento

CAMPINAS
2023

Jesamin Melissa Zevallos Quispe

UpKG: A Framework to Insert New Domains in Knowledge Graphs

UpKG: Um Framework para Inserir Novos Domínios em Grafos de Conhecimento

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Julio Cesar dos Reis

Este exemplar corresponde à versão final da Dissertação defendida por Jesamin Melissa Zevallos Quispe e orientada pelo Prof. Dr. Julio Cesar dos Reis.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Z61u

Zevallos Quispe, Jesamin Melissa, 1997-
UpKG a framework to insert new domains in knowledge graphs / Jesamin Melissa Zevallos Quispe. – Campinas, SP : [s.n.], 2023.

Orientador: Julio Cesar dos Reis.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Grafo de conhecimento. 2. Comércio eletrônico. 3. Web semântica. 4. Ontologias (Recuperação da informação). 5. Framework (Programa de computador). I. Reis, Julio Cesar dos, 1987-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações Complementares

Título em outro idioma: UpKG um framework para inserir novos domínios em grafos de conhecimento
Palavras-chave em inglês:
Knowledge graph
Electronic commerce
Semantic Web
Ontologies (Information retrieval)
Framework (Computer program)
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Julio Cesar dos Reis [Orientador]
Veruska Carretta Zamborlini
André Santanchè
Data de defesa: 29-11-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0006-6419-4694
- Currículo Lattes do autor: https://lattes.cnpq.br/8205783809953362

Jesamin Melissa Zevallos Quispe

UpKG: A Framework to Insert New Domains in Knowledge Graphs

UpKG: Um Framework para Inserir Novos Domínios em Grafos de Conhecimento

Campinas, 29 de novembro de 2023

Acknowledgements

First of all, I thank God for guiding me during these two years of study.

I thank my parents and grandparents for their unconditional love and support given all this time, despite the distance I always felt close and present at this stage of my life.

I am especially grateful to my advisor Prof. Dr. Julio Cesar dos Reis for giving me the opportunity to be his student. In addition, his help was very important to finish this research work, since with his knowledge and patience they showed me many times the way to follow with this M.Sc. dissertation.

I would like to thank in particular my friend, partner and boyfriend Percy Maldonado, for his love, patience and understanding at this stage of my life, for being my emotional support to carry out this research work.

I thank the Institute of Computing of the University of Campinas (IC-UNICAMP), for opening the doors and giving me the opportunity to advance in my professional career.

A special thanks to GoBots company and all its members who supported me with their knowledge.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001. This work was partially supported by the São Paulo Research Foundation (FAPESP) (Grant #2022/15816-5)[1].

[1]The opinions expressed in this work do not necessarily reflect those of the funding agencies.

Resumo

Nos últimos anos, a criação de Knowledge Graphs (KGs) tem avançado significativamente. Eles se tornaram essenciais em vários domínios, como o *e-commerce*. Aplicativos de *e-commerce* os aplicam na busca e recomendação de produtos, sistemas de questões e respostas e chatbots (assistentes) virtuais, entre outras tarefas. No entanto, o *e-commerce* deve cobrir constantemente novos domínios/ categorias para responder às novas necessidades dos usuários. Esse processo requer uma análise rigorosa para abranger novos domínios, adicionando novos conhecimentos não armazenados no KG em uso. Esta Dissertação de Mestrado propõe, constrói e avalia um *framework* que nomeamos **UpKG** para inserir novos domínios em KGs existentes dentro de um contexto de *e-commerce*. Nossa abordagem se baseia em perguntas e respostas coletadas. Nosso framework foi aplicado no contexto real da empresa GoBots para facilitar o processo de inserção de novos domínios, empresa especializada em soluções de e-commerce na América Latina. O caso realizado considerou um KG existente, utilizado para uma aplicação de questões e respostas no domínio Automotivo. No estudo de caso conduzido, nosso trabalho abrangeu e inseriu novas triplas relacionadas ao domínio de Eletrodomésticos, obtendo um KG que suporta o domínio Automotivo e Eletrodomésticos. O KG gerado foi preenchido por 3382 novas instâncias que foram extraídas de 1338 perguntas e respostas reais do banco de dados da GoBots. Realizamos dois tipos de avaliações: o primeiro focado na avaliação da ontologia, e a segunda para a avaliação do KG gerado. Para a avaliação da ontologia utilizamos três ferramentas: Reasoner Pellet, OOPS! e OntoDebug, que nos permitiu garantir a consistência e coerência da estrutura dos conceitos e propriedades modelados. Para a avaliação do KG, exploramos trinta Questões de Competência (15 do domínio Automotivo e 15 do domínio Eletrodomésticos). Isso permitiu avaliar se as instâncias atendem ao requisito da aplicação, que é responder a questões de compatibilidade entre produtos inseridos por usuários em plataformas de *e-commerce*. A avaliação realizada demonstrou a viabilidade e aplicabilidade do nosso framework UpKG em diferentes domínios no *e-commerce*. Nossa contribuição permite a expansão do KG para novos domínios.

Abstract

In recent years, the creation of Knowledge Graphs (KGs) has advanced significantly. They have become essential in several domains, such as e-commerce. E-commerce applications apply them in the search and recommendation of products, question-answering systems, and virtual chatbots, among other tasks. However, e-commerce must constantly cover new domains/categories to respond to new user needs. This process requires rigorous analysis to cover new domains, adding novel knowledge not stored in the KG under use. This MS.c Dissertation proposes, builds, and evaluates a framework we named **UpKG** to insert new domains in existing KGs within an e-commerce context. Our approach relies on questions and answers collected. We applied our study in the real-world context of the GoBots, company specialized in *e-commerce* solutions in Latin America, to facilitate the new domain insertion process in a KG. The conducted case study was on an existing KG, whose triples were dominated by the Automobile domain. Our investigation by applying the UpKG framework covered and inserted new triplets related to the Appliances domain, obtaining a KG that supports the Automobiles and Appliances domain. The final KG was populated by 3382 new instances, extracted from 1338 real-world questions and answers from the GoBots database. In addition, we carried out two types of evaluations: the first was focused on evaluating the ontology, and the second on evaluating the KG. To evaluate the ontology, we explored three tools: Reasoner Pellet, OOPS!, and OntoDebug, which allows us to ensure the consistency and coherence of the modeled ontology. To evaluate the KG, 30 Competency Questions were used (15 from the Automobiles domain and 15 from the Appliances domain), which allowed us to evaluate whether the instances align with the application, which is to answer compatibility questions. The evaluation showed the viability and applicability of the UpKG framework in different domains within e-commerce. Our contribution allows the expansion of the KG to new domains.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Context and Motivation

In recent years, the creation of Knowledge Graphs (KGs) has made significant progress. With the rapid growth of data in different fields, KGs play a vital role in transferring large amounts of data to actionable knowledge. They are essential in various domains, including e-commerce [22] [29] [11] because several applications related to online commerce explore them in different areas, such as product searches, product recommendations, and virtual chats, question answering, among others. Several big-player companies have used KGs in e-commerce, including Alibaba [22], Walmart [38], and Farfetch [4], among others.

Due to constant changes, e-commerce systems must be updated to continue providing better customer service. They must constantly cover new domains to increase knowledge of KG and update KGs, inserting knowledge through triples based on a new domain/category to respond to the novel users' needs. However, updating a KG involves expanding it and inserting domains such as appliances, furniture, and automobiles.

The insertion of new domains into a KG is not so trivial as it relates to the complexity of domain knowledge. Each domain, such as Clothing, Furniture, Automobiles, Appliances, etc. presents its terms, relationships, and features. Inserting a new domain is as important as understanding this domain and its specific terms to represent it in an ontology properly. A lack of understanding of the domain can lead to building incorrect ontology and not capturing key relationships and features of products. In addition, ensuring that information is represented in a consistent and standardized manner is very important because inconsistency may result in ambiguity and confusion in the interpretation of data. Therefore, A robust approach is needed to structure and standardize information consistently in ontology.

The literature presents research studies related to KGs in e-commerce [22] [11] [29]. Some studies explore the extension of domains in KGs [3, 31] and ways of populating triples in KGs [19] [29]. However, none of the analyzed studies explores these three critical aspects together of inserting new domains into e-commerce KGs, which are (i) KG in e-commerce, (ii) Populating KG, and (iii) Extending KG. The difference between Populating KG and Extending KG is that Extending KG is expanding the KG at the domain level and Populating KG is populating the KG with data in a specific domain.

Currently, this research is condicted in a real e-commerce scenario within the GoBots company, which presents solutions to answer user compatibility questions using a KG as can be seen in Figure 1.1.

1.2 Problem Characterization

E-commerce systems constantly evolve and seek to cover new domains to improve users' needs. This expansion of domains is essential to improve the user experience, increase sales, and provide high-quality responses, among other aspects. However, the need to cover new domains leads to the KG update. This task is not trivial, as it requires a thorough analysis of the KG structure and ontology to ensure proper integration of the new domain. We present two cases in which these challenges become evident in the following.

Case 1. Figure 1.1 shows a scenario in which a customer communicates with a chatbot of an online store that uses various solutions, including a KG. Initially, the customer queries about the compatibility of a product with his Ford Fiesta car, to which the chatbot responds affirmatively, generating customer satisfaction. However, the situation changes when the customer asks another question about a different domain. In this case, technology. The chatbot's response indicates that it "cannot understand" and generates frustration in the customer.

This case shows a knowledge gap in the online store, showing that while it can answer car-related questions, it cannot provide answers regarding technology questions. Addressing this deficiency is crucial to improving the quality of responses and customer satisfaction.

This case shows that the online store has a superior ability to handle automobile queries since its current KG has information in this domain. However, it shows a lack of knowledge regarding technology-related questions, as the existing KG does not cover this domain.

This lack of cross-domain knowledge directly impacts the quality of responses and customer satisfaction. This requires addressing the expansion of KG to include the domain related to technology, similar to how the domain of automobiles is handled. This might allow the chatbot to provide more complete and accurate answers in both domains, thus providing a more satisfying experience to users.

Case 2. Figure 1.2 presents two different KGs, one dedicated to technology and another focused on automobiles. KG online store shows the structure of the same with focus to the domain of Automobiles and the KG of the new identified domain shows the structure of the ontology with focus to the domain of Technology. If our goal is for the online store to answer questions from the technology domain, it means that the KG with the Automobiles domain must cover the Technology domain.

However, this process is not simple, as these two domains present properties and relationships that are not directly compatible. Therefore, the challenge must consider how adapting the automobile KG's ontology to host the technology domain effectively. This task is essential to ensure the consistency and integrity of the resulting KG.

On the other hand, we have another challenging aspect, which is the extraction and

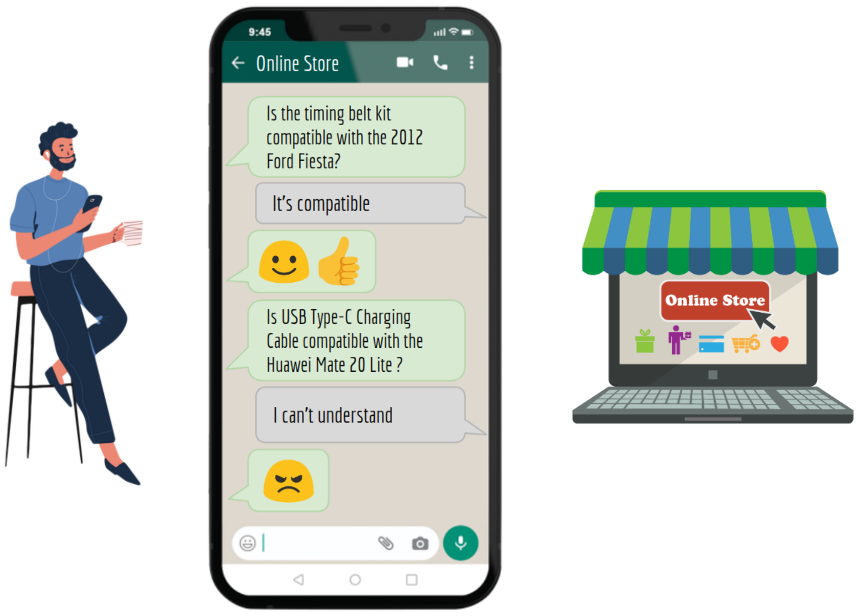

9:45
Online Store
Is the timing belt kit compatible with the 2012 Ford Fiesta?
It's compatible
Is USB Type-C Charging Cable compatible with the Huawei Mate 20 Lite ?
I can't understand
Online Store

Figure 1.1: Example in a chatbot problem

generation of information necessary to populate our KG, taking into account the newly inserted domain in addition to the existing domains. To generate such information, extracting intents and entities with a high confidence level is necessary according to the relationships in ontology. The entity represents a term with a relevant meaning to understand the context of the sentence. For example, “brand", “model", “year", are present in the Automobiles domain. Identifying entities is important as it allows understanding of questions related to this domain. The intent is to identify the intents of the question, whether the question refers to compatibility between entities, comparison between products, product information, etc.

Each new domain brings additional development to extract all the entities relevant to the domain, as is the case of new properties that were not mapped in previously existing domains. Extracting triplets from questions and answers is challenging due to the question variability, ambiguity in meaning, and the need to understand the context. For example, Figure 1.2 presents the need to recognize the property *size* and store it in the existing KG that only supports the domain of the automobile, where such property does not exist in the ontology of the KG.

To better visualize the problem to be solved, we propose two pairs of questions and answers in the following, one of them related to an existing domain in the KG (Automobiles) and the other related to a potential new domain (Technology):

- Automobile Domain.
  - Question: Is the product X compatible with the Nissan Qashqai 2012?
  - Answer: Hello, yes it is compatible.
- Technology Domain.
  - Question: Does Y work for the Huawei P30 Pro 6.47" cell phone?

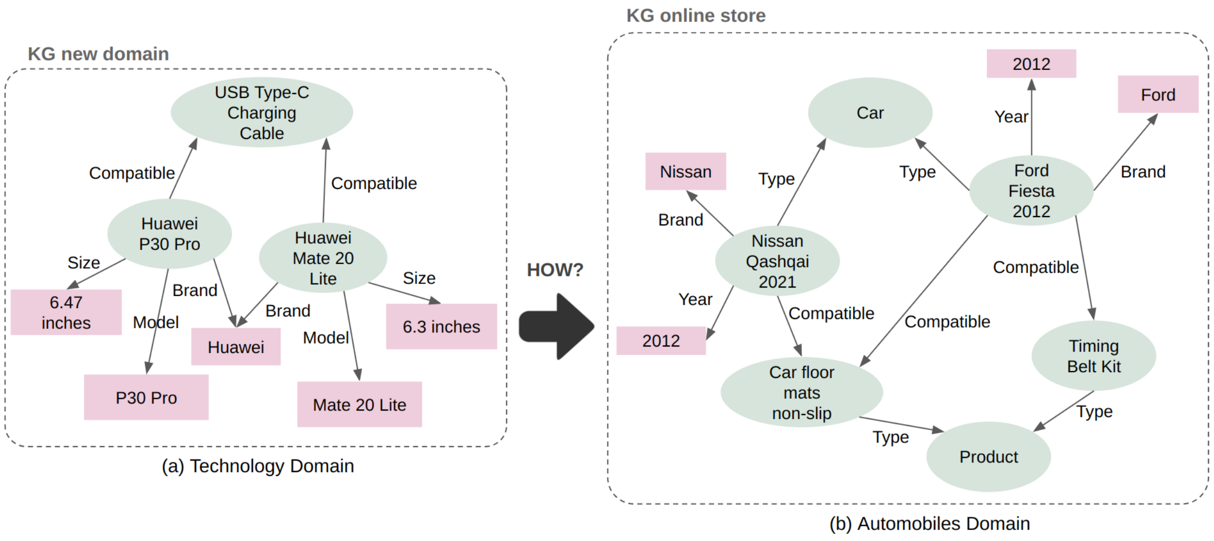


Figure 1.2: Example of domain insertion problem at the KGs level. (a) KG of the new Technology domain; (b)KG of the online store that covers the Automobiles domain.

– Answer: Good afternoon, unfortunately the product is not compatible with Huawei.

As the current KG is focused on questions and answers related to the Automobile domain, it is possible to extract the necessary information from the question with everything previously developed to populate the current KG, such as the brand, model, and year. However, the problem begins when we aim to analyze a question from a domain different from automobiles, for which we would have to be able to recognize all the new entities present in the question that are useful for our new KG.

From the second question, we can easily obtain the following triplets (not trivial for a computer to address it automatically). Of which, the *size* property cannot be inserted directly into the KG, and this would generate information loss.

- `<Huawei P30 Pro>, <brand>, <Huawei>`
- `<Huawei P30 Pro>, <model>, <P30>`
- `<Huawei P30 Pro>, <size>, <6.47">`
- `<Huawei P30 Pro>, <type>, <Technology>`
- `<Y>, <no-compatible>, <Huawei P30 Pro>`
- `<Y>, <type>, <Product>`
- `<Y>, <gtin>, <123456789>`

Our examples showed that adding a new property entails a restructuring of the existing KG regarding:

- Modeling (ontology) efficiently all the relationships of the new domain;
- Combining the new properties with the existing properties of the current domain;
- Extract new entities from the question (challenging);

- Generating triplets following the ontology developed to retrieve the information through SPARQL queries finally. A triple is an element `<s, p, o>` where “object o stands in relationship p with subject s". The first component s is called the subject of the triple; p is called the predicate; and o is called the object [15]

Inserting new domains into an existing KG is challenging due to the need to address ontological modeling, entity relating, conflict resolution, consistency across instances, scalability and performance, and evaluation of the ontology and the KG. Inserting new triples and relationships must be carefully managed to preserve the coherence and semantic integrity of the existing KG. We further detail each of these challenges in the following:

- **Ontological modeling:** KGs are usually based on ontologies that define the relationships and properties between entities. The introduction of a new domain involves the potential creation or extension of existing ontology classes and properties, which requires careful modeling to ensure consistency with existing knowledge structures.

  « Knowledge Graph stands for a graph of data intended to accumulate and convey real-world knowledge, where nodes represent entities of interest and edges represent different relationships between these entities [17]. »

  « Ontology [35] refers to the shared understanding of some domain of interest, which may be used as a unifying framework. Ontologies are formal models that define concepts, properties, and relationships in a specific domain. These ontologies provide a unified knowledge structure that improves the understanding and processing of information. »

- **Entity relationship:** The KG depends on the relationship between entities. Introducing a new domain involves establishing relationships between entities in the new domain and existing ones. Identifying and creating these relationships consistently can prove to be challenging.

- **Conflict resolution:** Inserting new domains can lead to modeling conflicts when properties or terms used in the new domain conflict with existing ones. Resolving these conflicts without compromising the coherence of the ontology is crucial.

- **Consistency in instances:** Consistency in instances is essential. Inserting new data into the KG must ensure that the instances comply with the ontological constraints and do not lead to inconsistencies.

On the other hand, it is crucial to highlight that the incorporation of a new domain in KG arises as a solution to the current problem in the GoBots company, which has a specific KG for a given task and domain. The identification of this limitation was made through internal GoBots metrics designed to evaluate the tools that support the answers to questions asked by users in an e-commerce environment.

Similarly, we note that the main purpose of the current GoBots KG is to answer questions that are intended for compatibility. For this reason, the application of our proposal in Chapter 4 is focused on a KG that reflects knowledge related to compatibility.

1.3 Objectives and Research Questions

In this M.Sc. Dissertation, we propose the **UpKG** framework, which allows inserting new domains in an existing KG in an e-commerce context based on the input from users' questions and answers. Our study aims to attain the following specific objectives:

- Conceptually develop UpKG focusing on e-commerce context based on users' natural language questions and answers as one type of data input.
- Evaluate the proposed UpKG framework in the real-world context of a company;
- Ensure that our framework is flexible enough to add new domains focusing on e-commerce.

We present two research questions that drive our research in this Dissertation.

1. How can the UpKG framework can be suited to the constant information evolution in e-commerce, allowing the insertion of new domains without compromising the coherence and integrity of the underlying KG in place?
2. How does inserting new KG domains through our proposed UpKG framework impact end-user queries on e-commerce platforms?

1.4 Synthesis of Our Methodology and Findings

Figure 1.3 presents the synthesis of our research methodology and how this is connected to the Chapters in this Dissertation as follows:

- **Understanding the literature (Figure 1.3 (a)).** We reviewed essential concepts to comprehend the scope of our research. We studied fundamental terms such as Triples, Ontologies, Knowledge Graphs (KG), RDF, and SPARQL, which are essential to understanding the foundations of our study. In addition, we investigated previous studies focusing on research directly linked to our challenges. We observed the weaknesses presented in the state of the art in expanding domains within a KG. Chapter 2 presents the results of this step.
- **Proposing the UpKG framework (Figure 1.3 (b)).** At this stage, we design our UpKG framework, which focuses on inserting new domains into an existing Knowledge Graph (KG), focusing on e-commerce using Natural language questions and answers from users as input. The framework is composed of four modules: (i) Understand the current KG; (ii) Identify new domains; (iii) Restructure the ontology; and (iv) Populate the KG. UpKG provides a step-by-step guide to expanding a KG consistently and efficiently and can adapt to a variety of domains, with a focus on e-commerce. Chapter 3 presents the results of this step with all the modules within the framework.

1. **Understand the current KG.** The structure of the existing KG is analyzed in detail, including its relationships and ontology. This module provides a solid foundation for understanding how the current KG is built.
2. **Identify new domains.** The goal is to find new domains where the current KG cannot answer questions. This is achieved by identifying unanswered questions, and unmapped relationships and determining the domain with the largest number of unanswered questions.
3. **Restructure the ontology**. This emphasis is on modifying the existing ontology to adapt the new relationships related to the domain identified in the previous module. This involves selecting a base ontology and combining properties mapped to the current ontology.
4. **Populate the KG.** This aims to insert triplets into the current KG. This involves extracting intents and entities from the questions and answers, followed by creating RDF triples that are stored in an RDF store.

- **Applying the framework in a Case Study in an Industrial Setting (Figure 1.3 (c)).** At this stage of our methodology, we apply the implementation of the UpKG framework in a real scenario. The application context is in the company GoBots[1], where UpKG was applied to add a new domain to their existing KG. Our obtained results expand the KG's capacity to answer questions in different domains and improve the user experience in e-commerce. The process of applying the UpKG framework reached the following results:

  1. Understood the current GoBots KG in place (in production), its structure, properties and relationships.
  2. Identified a new domain, in this case it was the “Appliances" domain.
  3. Restructured the ontology to allow the inclusion of the new domain.
  4. Populated the KG with questions and answers related to the Automobiles and Appliances domains.
  5. The final result was creating a new version of the KG considering further 1338 question and answer pairs from the Automobiles and Appliances domains.

- **Evaluating the generated Assets from the Application (Figure 1.3 (d)).** At this step, we evaluated two key assets generated in our research: the ontology and the Knowledge Graph (KG).

  1. For the ontology evaluation, we applied three methods of validation: Reasoner Pellet [2, 20, 26], OntoDebug [34, 21, 26], and OntOlogy Pitfall Scanner (OOPS!) [37, 36, 34, 14, 26]. We decided to apply these three validation methods due to being present in the validation of several studies in the literature. In addition, all three methods contribute to a more complete and accurate

[1]`www.gobots.ai` – This study is under the context of a formal Partnership between UNICAMP and GoBots.

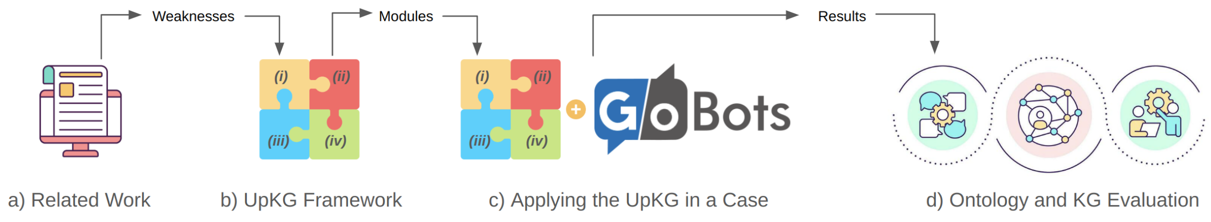


Figure 1.3: Overall research methodology

validation of the ontology. Reasoner Pellet is an inference engine and allows to infer new information from the ontology, it is also used to check the general consistency of the same ontology. OntoDebug is a plugin that allows to identify inconsistent axioms of greater complexity which could be difficult for the reasoner to identify. OntoDebug allows you to identify whether the ontology is coherent and consistent in structure and logic. OOPS! allows you to check the ontology online and provides a detailed list of common ontology pitfalls. This tool helps identify and correct possible pitfalls, improving the quality of the ontology. Our results showed that the initial ontology presents inconsistencies and errors. We solved it in a second iteration, resulting in a coherent ontology. Overall, these evaluations were crucial to ensure the ontology's and KG's quality and usefulness for the real-world applications.

2. To evaluate the KG, Competency Questions (CQs) were used, which aimed to assess to which extent the generated KG can provide accurate answers to questions related to a specific domain.

1.5 Dissertation Organization

This M.Sc. Dissertation is organized as follows.

Chapter 2 presents the fundamental concepts that serve as the basis for our research work. We present the related studies that are relevant topics for our research. We carry out a comparison in three areas of studies related to our study: (i) **KG in e-commerce**, we reviewed the existing literature on KGs present in e-commerce; (ii) **Extending KG**, we reviewed the existing literature on extending domains in a KG; and (iii) **Populating KG**, we reviewed the existing literature on methods or approaches used for popular a KG.

Chapter 3 describes in detail our proposed UpKG framework. UpKG framework aims to insert new domains into an existing KG, focusing on e-commerce from existing users' questions and answers. We details its four modules: (i) Understand the current KG; (ii) Identify a new domain; (iii) Restructure the ontology; (iv)Populate the KG. Each module consists of a series of steps that represent the processes necessary for the module to reach its objective. These steps are described in detail to provide a complete understanding of their functionality and contribution to each module.

Chapter 4 presents the application of the UpKG framework in the context of GoBots, which offers solutions to structure the knowledge of compatibility between products. This Chapter shows the results of applying our UpKG framework in each designed module. We present our obtained findings in each step that composes a module, describing the development process and the results of each module of our framework. The documented details of our Case Study provide a complete overview of the application process of our UpKG framework.

Chapter 5 describes the methods and results for assessing the final outcomes (artifacts) of applying the UpKG framework. In particular, we demonstrate how the obtained ontology and the entire KG are soundness.

Chapter 6 presents our contributions and conclusions from our research work. In addition, we describe planned future studies.

Chapter 2

Literature Review: Background and Related Studies

This Chapter aims to provide a background for understanding the context of our research work.

Section 2.1 describes the fundamental concepts essential to understanding our study proposal. These terms include, among others, fundamental concepts such as Triples, Ontology, Knowledge Graph(KG), RDF, and SPARQL.

Section 2.2, describes the explored tools in our research. The tools used in our research are: (i) Protégé; allows to visualize and edit ontologies; (ii)Reasoner Pellet; allows to infer new information from ontology; (iii)OntoDebug; allows to identify inconsistent axioms of greater complexity; and (iv)OOPS!; provides a detailed list of common pitfalls in ontology.

Section 2.3, describes related studies directly linked to our study. In this section, an analysis of studies is carried out, which provides a foundation and literature positioning for our research. The related studied focused on three important scopes connected to our study: (i) KG in e-commerce, the application of KG in an e-commerce context; (ii) Extending KG, the modification of the KG by adding new instances and relationships to cover new domains of knowledge; and (iii) **Populate KG**, the addition of new knowledge in the KG based on the structure defined by the ontology. Our study aims to address these three scopes to expand the reach and effectiveness of the KG in new domains within the e-commerce environment.

2.1 Fundamental Concepts

In inserting new domains within an existing KG, the following concepts are intrinsically interconnected and play a crucial role in the representation, storage, retrieval, and evaluation of knowledge.

Triples. Triples are essential elements in the representation of information within a KG. These triplets are used to express semantic relationships between different elements. Formally, a triple is an element `<s, p, o>`, typically interpreted as a statement where "object o stands in relationship p with subject s". The first component s is called the subject of the triple; p is called the predicate; and o is called the object [15].

The triple components are used to express semantic relationships between different elements of information. The subject represents an entity or resource, the predicate denotes a property or relationship that links the subject, and the object represents the value or description associated with that property. The triples are the basis of the Knowledge Graph (KG) and allow the representation of data in a semantic and structured way.

Figure 2.1 shows the structure of a triple.

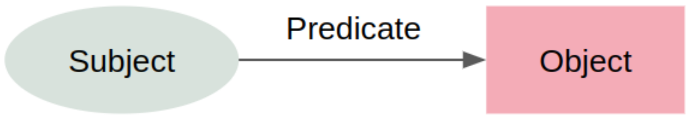


Figure 2.1: Structure of a triple

Resource Description Framework (RDF). Triples are combined to build RDF, a standard that provides a framework for semantic data representation. RDF uses triples to describe resources and their relationships. RDF [12] is a fundamental standard in the Semantic Web [13] used to represent and model data in a way that is understandable to humans and machines.

RDF aims to provide a framework for describing relationships between web resources through triples, consisting of a subject, a predicate, and an object. These triples combine to form KGs, allowing the representation of Linked Data [7], which facilitates interoperability and reasoning on information. In addition, RDF is used to represent and exchange data on the Web, allowing machines to understand information more meaningfully.

Ontology. According to Uschold and Gruninger [35], Ontology refers to the shared understanding of some domain of interest, which may be used as a unifying framework. Ontologies are formal models that define concepts, properties, and relationships in a specific domain. These ontologies provide a unified knowledge structure that improves the understanding and processing of information. An ontology necessarily entails or embodies some sort of worldview with respect to a given domain. The worldview is often conceived as a set of concepts (*e.g.*, entities, attributes, processes), their definitions, and their interrelationships. This is referred to as a conceptualization. Therefore, ontology is a semantic model syntactically defined that models the things that exist in our domain and models general things that have common characteristics.

Ontologies [6] are used to standardize and give meaning to data, facilitating interoperability and understanding of information. An ontology is structured around a hierarchy of classes and properties, where classes represent concepts or types of entities, and properties define the relationships between these entities. In addition, ontology plays an important role in data search, reasoning, and enrichment, which enables applications to understand and use information effectively.

Web Ontology Language(OWL). OWL is a standard that allows the representation and definition of ontologies in the Semantic Web, being essential to model concepts, properties and relationships. OWL [24] is an ontology language developed as a World Wide Web Consortium (W3C) standard for ontology representation on the Semantic

Web. OWL is used to define and represent knowledge in a formal and structured way in various domains, which allows modeling concepts, properties, and relationships within a specific domain.

The elements included in OWL are classes, properties, individuals, and restrictions. Classes are used to define concepts or categories in the domain, properties describe relationships between classes or individuals, individuals represent specific instances of classes, and constraints define rules that maintain the structure and behavior of ontology. OWL captures the semantics and structure of knowledge in a format that is interpretable by both humans and machines. OWL has an important role in the representation of ontologies and in the creation of knowledge systems in the Semantic Web, which allows greater efficiency in the search, integration, and processing of information.

Knowledge Graph (KG). Ontologies and RDF data form a Knowledge Graph, which organizes and stores information semantically. KG stands for a graph of data intended to accumulate and convey real-world knowledge, where nodes represent entities of interest and edges represent different relationships between these entities [17]. A KG is the representation of knowledge based on an ontology. Therefore, a KG is created when they apply an ontology to a dataset (cf. Figure 2.2).

Figure 2.2 shows how ontology and data are related to compose a KG, where the data is integrated into the KG following the definitions and structures established by the ontology. Each entity and relationship in the KG is supported by concepts defined in the ontology. Therefore, the ontology establishes the rules and conceptual structure, the data conforms to these rules and the KG visualizes and organizes this information graphically, creating relationships that facilitate the representation and analysis of knowledge.

Representing information in the form of a graph allows the precise capture of the relations and semantic connections between the data, which facilitates the search and reasoning of the information. KGs are important because they allow a structured representation of knowledge and improve information retrieval and automatic inferences.

Some key KGs entities are Google [1] and DBpedia [2]. These KGs are essential to organize and enrich knowledge on the web.

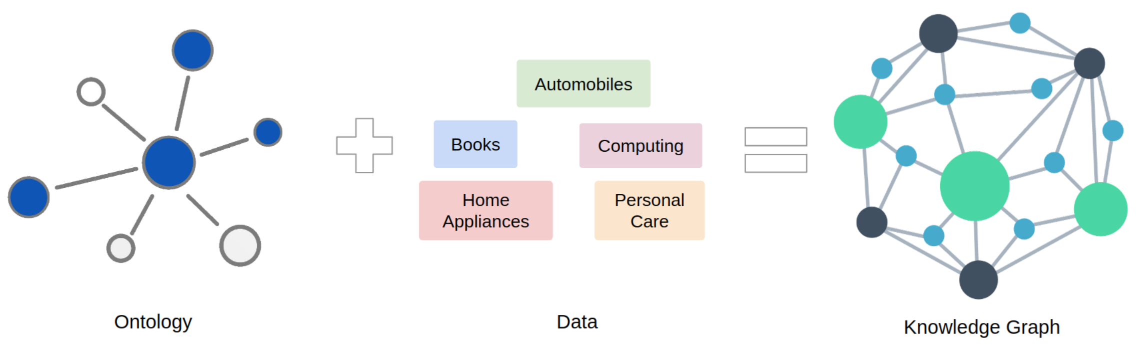


Figure 2.2: Representation of the composition of Knowledge Graph

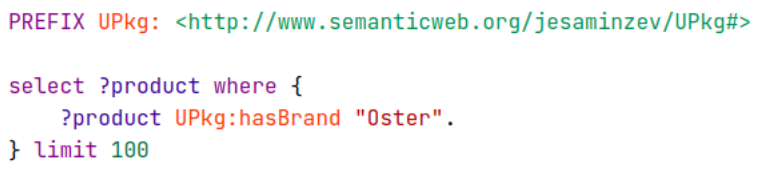
SPARQL Protocol and RDF Query Language (SPARQL). SPARQL [28] is a

[1]https://developers.google.com/knowledge-graph/
[2]https://www.dbpedia.org/

query language that aims to retrieve and manipulate data stored in RDF format, which facilitates searching and retrieving information in databases and KGs. SPARQL allows flexible querying; it can also search for and extract specific information from a dataset, essential for building a web where data is interconnected and enriched with semantic meaning.

SPARQL has a syntax similar to the Structured Query Language(SQL) but it was specifically designed for semantic data querying [23]. SPARQL queries can search for patterns in RDF graphs, filter data, join multiple data sets, and perform complex operations on data, making it an essential tool for extracting meaningful knowledge and relationships on the Semantic Web.

Figure 2.3 shows an example of a SPARQL query, which seeks to retrieve up to 100 products that are brand “Oster" in an RDF dataset. The UPkg prefix was defined to abbreviate long URLs and facilitate query writing. In addition, the query looks for resources (in this case, products) that have a property called “‘UPkg:hasBrand" with the value “Oster" and limits the result to a maximum of 100, and avoids the query returns too large a result set.



```
PREFIX UPkg: <http://www.semanticweb.org/jesaminzev/UPkg#>

select ?product where {
    ?product UPkg:hasBrand "Oster".
} limit 100
```

Figure 2.3: Example of a SPARQL query that aims to return products that have the Oster brand

Universal Resource Identifiers (URI). Resources in the Semantic Web are identified by URIs, which provide a unique schema for resource identification, including concepts in ontologies and triples in RDF.

URI [5] is a central component in the Semantic Web that provides a unique and uniform identification for resources on the World Wide Web. In addition, the URIs aim to identify resources on the Web uniquely and uniformly; their initial purpose is to constitute a universal standard that enables the identification of various resources, such as web pages, documents, images, web services, videos, and any other elements present on the Internet.

URIs can take various forms, such as Uniform Resource Locators (URLs) that are used to locate specific resources on the Web, and Uniform Resource Names (URNs) that are used to name resources regardless of their physical location. Figure 2.4 presents an example of the URI, URL, and URN structure.

URIs are important for hyperlink operation, web browsing, and Semantic Web data interconnection. For this reason, URIs have an important role in constructing the Semantic Web because they allow different sources of information to be linked and related in a significant way, which contributes to the creation of an interconnected knowledge graph rich in knowledge.

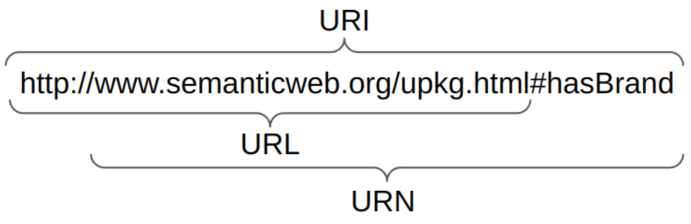


Figure 2.4: Example of URI, URL and URN

2.2 Explored Tools

Protégé. Protégé [25] is a popular and widely used knowledge modeling software tool developed at Stanford University. This tool offers functionalities for the modeling and management of ontologies in different formats and facilitates the creation, visualization, and manipulation of ontologies efficiently. In addition, Protégé supports framework-based ontologies and follows the Open Knowledge Base Connectivity (OKBC) protocol [10] to ensure data interoperability.

Protégé has an intuitive user interface that allows you to define classes, properties, instances, and relationships within an ontology visually. This tool supports multiple ontology languages, including OWL, which is important for knowledge representation. Protégé allows users to provide logical reasoning to infer new knowledge from relationships and constraints defined in an ontology. In addition, it offers a variety of plugins that expand its functionality and allow addressing different requirements and scenarios.

A reasoner like Pellet allows logical reasoning in ontologies, which enriches the ability of applications to understand and use knowledge. To ensure the quality of ontologies, OntoDebug and OntOlogy Pitfall Scanner! are debugging tools that detect problems and ensure ontological consistency. Our study explored these instruments as a way of assessing our generated ontology as a result of this investigation.

Reasoner Pellet. Pellet [32] is an open-source Java-based reasoner developed by Mind Swap group. It is based on the tableaux algorithm and supports expressive description logics. In addition, Pellet reasons ontologies through Jena as well as OWL-API interfaces and supports the explanation of bugs. Pellet provides various interfaces, including a command-line interface, an interactive Web form for zero-install use, DIG server implementation, and API bindings for RDF/OWL toolkits Jena and Manchester OWL-API.

The Reasoner Pellet acts as an engine that enriches the understanding of data by discovering new relationships, inferring properties, and validating compliance with restrictions defined in an ontology. This tool was designed to manage complex ontologies and works by processing RDF triples. In addition, it uses logical rules and reasoning algorithms to infer additional information. It is optimized for efficient and scalable performance, which is important for semantic web applications involving large data sets.

OntoDebug. OntoDebug [30] is a Protégé plugin that implements an interactive approach to ontology debugging; by having a faulty ontology, the tool finds a set of faulty axioms that explain the problem. Through interactive ontological debugging, the ontodebug [1] plugin helps detect and identify axioms that cause inconsistent or incoherency ontological mismatches.

Interactive ontological debugging is achieved by raising iterative questions in the form of axioms that the ontological engineer must answer. A query can be read as a question: Should the axioms given be included in the ontology?. This iterative process reduces the set of possible false axioms until the final set is identified. The plugin provides a repair interface to help fix axioms.

OntOlogy Pitfall Scanner! (OOPS!). OOPS! [27] is a web-based tool for detecting ontological problems that could lead to modeling errors. This software tool aims to help ontology developers detect errors and provides mechanisms to diagnose the forty errors described in the OOPS! pitfall catalog. OOPS! [3] has a Web page interface in which the user enters the URI of the ontology or its code. Once the ontology is parsed using the Jena API, the Pitfall Scanner inspects the declared ontology, looking for pitfalls among those available in the catalog. This software tool has the pitfall catalog and ontology as input data, and as a result, a list of pitfalls is obtained.

This tool applies predefined rules and heuristics to identify problems in ontology. *OOPS!* provides detailed descriptions or suggests corrections to improve ontology quality once possible problems have been identified.

2.3 Related Work

There are several studies related to the use of KGs for e-commerce. Some of them are based on specific tasks to insert a new domain.

KG in E-commerce context. Li *et al.* [22] proposed AlimeKG, an e-commerce KG representing knowledge about user problems, item information, and item relationships. AlimeKG helps to understand users' needs, answer pre-sales questions, and generate explanatory texts. This KG was applied to various business scenarios, such as buying guidance, answering property questions, and generating recommendation reasons. In addition, Li *et al.* [22] described how the domain KG comprises free text and tested it with various applications in e-commerce categories.

Xu *et al.* [38] proposed an approach based on a Product Knowledge Graph (PKG) to learn the inherent relationships between products, in which they make use of a method of learning distributed representation enhanced with self-attention. However, this study does not consider a customer's information as part of e-commerce because it is focused on products and how their information can be used for recommendation and classification tasks.

Populating KG. The T2KG [19] creates an automatic KG from natural language texts and fills an existing KG with new knowledge. Natural language context entities are assigned to the corresponding uniform resource identifier in the KG, which is usually the subject or object of the triples. They combined rule-based and similarity-based techniques to map the predicate of a triple generated from text in an existing KG. The experimental

[3] `https://oops.linkeddata.es/index.jsp`

part showed that the T2KG framework can successfully generate a KG and populate an existing one with new text knowledge. However, the framework does not work correctly when mapping predicates containing many compound words since the triplet extraction step is not perfect due to the complexity of the text in open domains.

Sant'Anna *et al.* [29] proposed a method to populate a KG from a collection of pairs of questions and answers, in which they performed an extraction of entities and intents to generate triples supported by the ontology defined in its KG. The knowledge generated and obtained from the pairs of questions and answers is stored in the KG. However, the method requires that the attendants answer the questions correctly through a manual process. Similarly to our work, the method proposed by Sant'Anna *et al.* [29] focused on the e-commerce domain.

Integroly [16] is a framework that automates the KG populating. It collects data from digital media sources to populate a KG focusing on the political marketing domain. They used a set of Natural Language Processing (NLP) techniques to extract knowledge from political marketing texts written in Spanish. In their experiments, the authors used Twitter and political news. They found problems with synonyms, ambiguities, and the connections made in the KG.

Extending KG. KBot [3] is a multilingual chatbot that understands user queries. KBot has automated learning based on classifying intents. In addition, KBot can add a new dataset to the existing knowledge base, allowing the expansion of the chatbot's capabilities and thus having a greater understanding of queries.

KGs have many applications and are important in the medical sector. It is challenging to build a KG for all diseases. For this reason, DEKGB [31] proposed an efficient and extensible framework to build KG for specific diseases based on doctors' knowledge. They described the process by extending an existing health KG to include a new disease.

Table 2.1 shows the organization and conceptual analysis we performed for the existing exploratory literature review conducted.

We considered three scopes. Our study aims to cover these three scopes: (i) KG in e-commerce, use of KG in an e-commerce platform; (ii) Populating KG, populating a KG with new knowledge based on the ontology; and (iii) Extending KG, modifying the KG by adding new instances with their relations. The goal is to build our framework to insert new domains in a KG with an e-commerce context based on users' questions and answers from GoBots, which owns a set of objects that contain the user's question and the human assistant's answer.

In our proposal, we base our research on [29], focusing specifically on the stage of populating KG. We apply the conceptual process proposed by this author in our framework. This approach allows us to take advantage of the methodologies and strategies presented in their work, adapting them to the needs and objectives of our framework.

2.4 Final Remarks

The Chapter aimed to present the background context of our research, where we present the fundamental concepts to understand our proposal and evaluations conducted and

Table 2.1: Comparison in three scopes of studies in KG connected to our present investigation in this article.

	KG in e-commerce	**Extending KG**	**Populating KG**
[19]			X
[31]		X	
[38]	X		
[3]		X	
[22]	X		
[29]	X		X
[16]			X
Our proposal	**X**	**X**	**X**

review the investigations related to our study. The Chapter aimed to establish a solid basis for our research work. The following Chapter focuses on a detailed description of our proposed framework, considering each module that makes up our framework, as well as a description of the steps involved in each of these modules, providing a complete overview of our UpKG framework.

Chapter 3

Our Proposed UpKG Framework

This Chapter provides a detailed description of our UpKG framework. We propose a framework to insert new domains in an existing KG focused on e-commerce from questions and answers. Figure 3.1 presents the overview of our UpKG framework. This consists of four modules: (i) Understand the current KG; (ii) Identify new domain; (iii) Restructure the ontology; (iv) Populate the KG. This Chapter is organized into four main sections, devoted to describing each module of our framework.

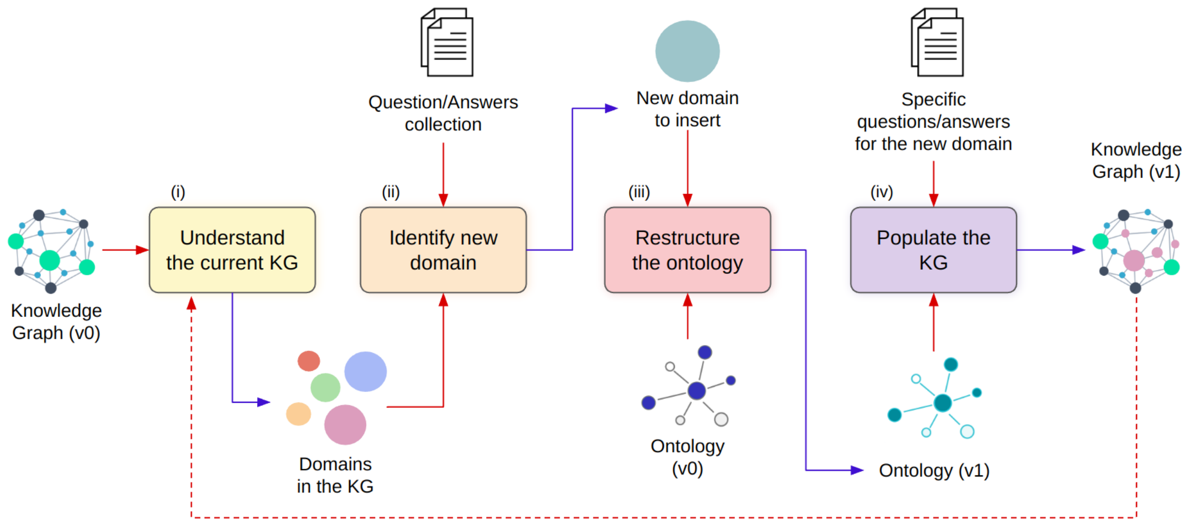


Figure 3.1: Overview of the UpKG Framework. (i) Understand the current KG: Module focused on understanding the current KG and its base ontology; (ii) Identify new domain: Module focused on discovering the new domain that will be inserted; (iii) Restructure the ontology: Critical module in which the base ontology is combined with the new properties; (iv) Populate the KG: Final module focused on generating the triples to insert the KG

Section 3.1 is focused on understanding the existing KG. This module involves how we propose a thorough analysis of the structure of a KG, considering the analysis of relationships and the structure of ontology.

Section 3.2 is focused on identifying new domains that a KG in place is not able to cover. This module consists of identifying questions that cannot be answered by the KG and identifying domains with a high incidence of unanswered questions. In addition, relationships that were not mapped in the ontology should be identified for this new domain.

Section 3.3 is focused on restructuring existing ontology to cover the new domain identified in Section 3.2. This module consists of selecting a base ontology that is related to the new domain to be inserted and combining mapped properties with the existing ontology. In this way, the creation of a final ontology is obtained that comprehensively covers the new domain previously identified in Section 3.2.

Section 3.4 is focused on inserting triples to a KG from existing questions and answers collected from users. This module includes extracting intentions and entities from questions and answers related to the domain previously identified in Section 3.2. In addition, it is carried out the creation of RDF triples that are later inserted in an RDF Store, allowing the construction of a populated KG that significantly covers the new domain.

3.1 Understand the current KG in place

Our framework works on an existing KG where the module at this stage aims to understand the shape and structure of the initial KG. Figure 3.2 shows the flow of steps involved in this module. Understanding the KG structure is necessary as our first step, which allows us to identify and analyze how the existing KG is structured and constructed, which we denote as `v0`.

A person with knowledge of ontologies can perform this step by observing data properties and attributes. For the second step, we analyze the structure of the existing ontology, which represents all the existing relationships in the KG. The comprehension of the ontology is as critical as that of the KG. In this step, one can use some tools to understand the ontology structure; one of the most popular tools of ontology visualization is Protégé [33]. This understanding is realized through the visualization and manipulation of ontology, for example we can use Protégé to make modifications and understand how ontology was developed, in the same way we can deepen and understand of the best way the relations between the entities that has the ontology.

In the third step, the relationships of the KGs are identified; this consists of observing all the possible relationships that the ontology allows and how it helps achieve the objective of the KG (e.g., answering users' questions).

The fourth step is to identify the domains currently supported by the KG `v0` and thus map the domains that are necessary to insert. In this step, one can access the current KG information and discern which domains the KG covers. At the end of the third step we have a list of all the ontology relationships and we can infer the domain that covers the KG thanks to the set of relations and properties obtained in the previous steps.

The development of this module can be performed manually since all the processes described require the intervention of human reasoning.

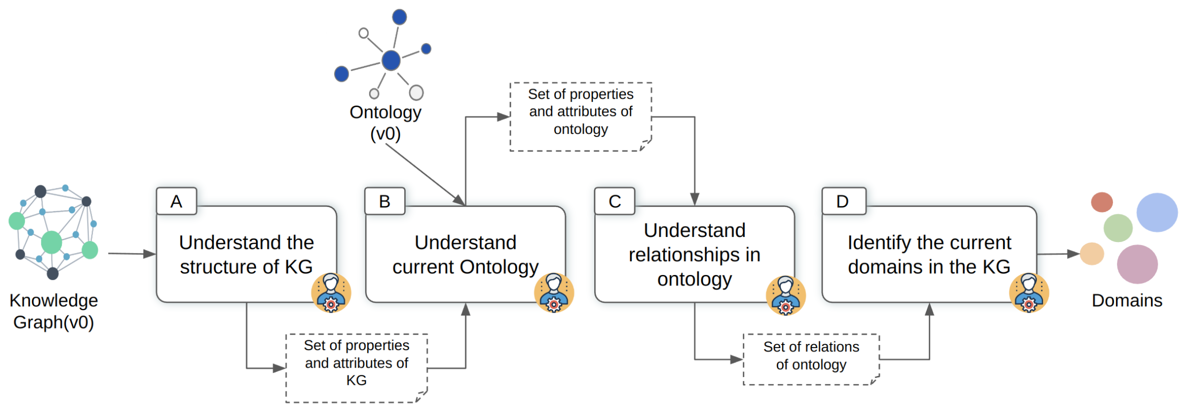

Ontology
(v0)
Set of properties and attributes of ontology
Knowledge Graph(v0)
A
Understand the structure of KG
B
Understand current Ontology
C
Understand relationships in ontology
D
Identify the current domains in the KG
Domains
Set of properties and attributes of KG
Set of relations of ontology

Figure 3.2: Flow of steps to understand the current KG in place. (A) Understand the structure of KG: First step focused on identifying the structure of KG, at the bottom right shows a man's icon this means that this step can be performed manually; (B) Understanding current ontology: The second step focused on understanding the structure of ontology; (C) Understanding relationships in the ontology: The third step focused on identifying the relationships that composes the ontology; (D) Identify the current domains in the KG: Fourth step focused on identify the domains covered by the KG.

3.2 Identify new domain

The objective of the second module is to identify a new domain where the KG in place cannot perform its task, in this case, answering questions that are not mapped in the current KG. Figure 3.3 shows the flow of steps involved in this second module.

The first step is to identify all the compatibility questions not answered by the existing KG. This step can be performed automatically through a script, which has as its input a collection of user questions and answers. The script has the function of obtaining the total number of questions for each main category or domain, this can be developed according to how the clients' questions are stored together with their respective answers by human assistants, in other words we can say that this step consists of grouping the number of questions for each category, excluding the domains covered by the current KG. A person with programming knowledge can execute this script supported by a query language or a programming language.

The second step identifies relationships that the ontology `v0` cannot map, making it impossible to respond to the KG according to its intent.

The third step identifies the domain with the highest number of unanswered questions from the current KG. To this end, a solution as an automatic script can generate a histogram of the main categories with the number of questions associated with each category. The resulting histogram shows the root categories with the number of unanswered questions, this histogram helps us to visualize the domain that has the largest number of unanswered questions and thus identify the new domain to be inserted in the KG. It is important to note that the histogram does not take into account the domain or domains covered by the current KG

This histogram allows a graphical presentation of domains not covering the current

KG; one can decide which domain to include based on the analysis of the histogram generated.

This module can be developed fully automatically, unlike the previous module (cf. Subsection 3.1), because each step can be applied using a software script.

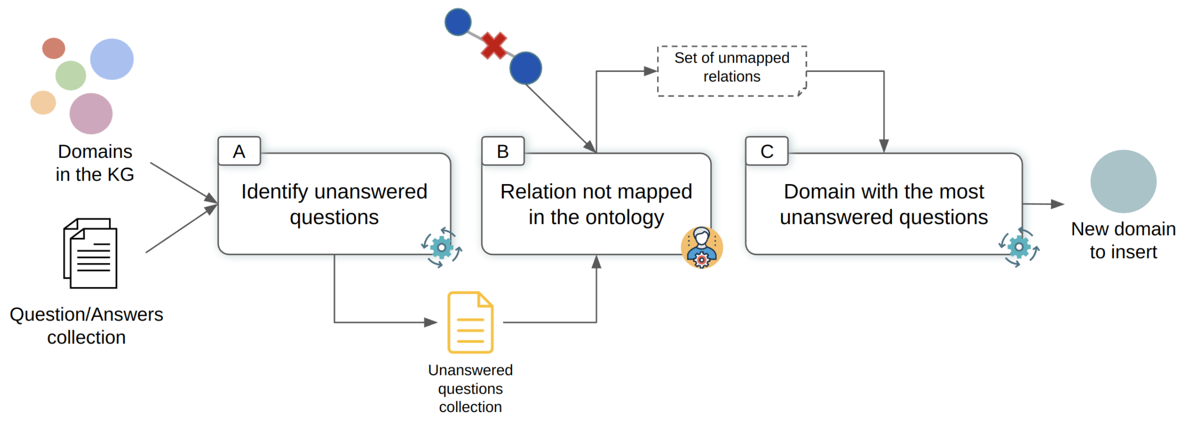

Domains in the KG
Question/Answers collection
A
Identify unanswered questions
Unanswered questions collection
B
Relation not mapped in the ontology
Set of unmapped relations
C
Domain with the most unanswered questions
New domain to insert

Figure 3.3: Flow of steps to identify new domain. (A) Identify unanswered questions: The first step focused on identifying questions not supported by KG, at the bottom right shows a blue icon this means that this step can be done automatically; (B) Relation not mapped in the ontology: The second step focused on identifying relationships not mapped by ontology; (C) Domain with the most unanswered questions: The third step focused on identifying the domain with the highest number of questions not answered by KG.

3.3 Restructure the ontology

The goal of the third module is to restructure the existing ontology with the relationships not mapped to the domain found in the previous module (Section3.2).

The input data to be used in this module is the new domain to be inserted and the existing ontology denoted as `v0`. Usually, an ontology is modified based on the new needs of a system. Therefore, it is essential to modify the structure of ontology and this process is done through the manual modification of ontology, using tools that facilitate the visualization and editing of classes and properties. Protégé is an important tool for visualizing and editing ontologies.

The data from this module is the ontology with the new domain to which we denote it by the version `v1`. Figure 3.5 shows the flow of steps involved in this third module.

The first step in this process is to search for available ontologies in the selected domain to reference to understand how this knowledge is structured. This search can be carried out through a literature review or searching in ontology repositories available on the Web. The second step consists in selecting a base ontology which becomes the starting point for modeling the new ontology. For example, we can select *Schema.org*, which presents properties widely used in different areas of knowledge, such as Products, Health, Persons, etc.

The third step aims to obtain the new ontology properties necessary for the new domain(s) identified and the type of questions to ask. For example, if our current domain is Automobiles, we could not answer questions about Technology. In this step, the intervention of human knowledge is necessary to combine the existing properties with the new ones; it is possible to find new properties necessary to store knowledge based on the set of questions and answers. For example, Figure 3.4 presents the analysis based on the recognition of entities and intentions in a question and answer pair. This analysis allows to identify the existing properties in the question, in this case we have properties such as *Brand*, *Model* and *Size*. The *Brand* and *Model* properties already exist in Automobile ontology. However, the *Size* property is not present, which leads to restructuring the Automobiles ontology to include the *Size* property. In addition, this question and answer analysis allows to identify the type of question intention, the product, and the type of compatibility of the answer.

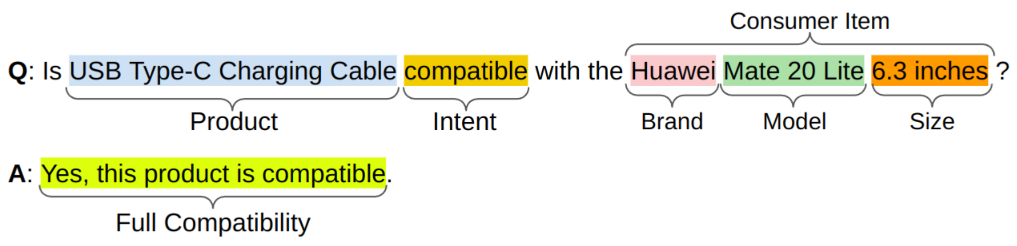

Consumer Item
Q: Is USB Type-C Charging Cable compatible with the Huawei Mate 20 Lite 6.3 inches ?
Product
Intent
Brand
Model
Size
A: Yes, this product is compatible.
Full Compatibility

Figure 3.4: Analysis of a question and answer pair from the Technology domain.

The fourth step manually combines the properties that were previously mapped with our ontology `v0`. In this step, one can use a software tool that facilitates ontology modeling (e.g. Protégé [1]) to obtain our ontology `v1` as output.

This module can be developed manually because, in all the steps, human intervention is necessary to restructure the ontology.

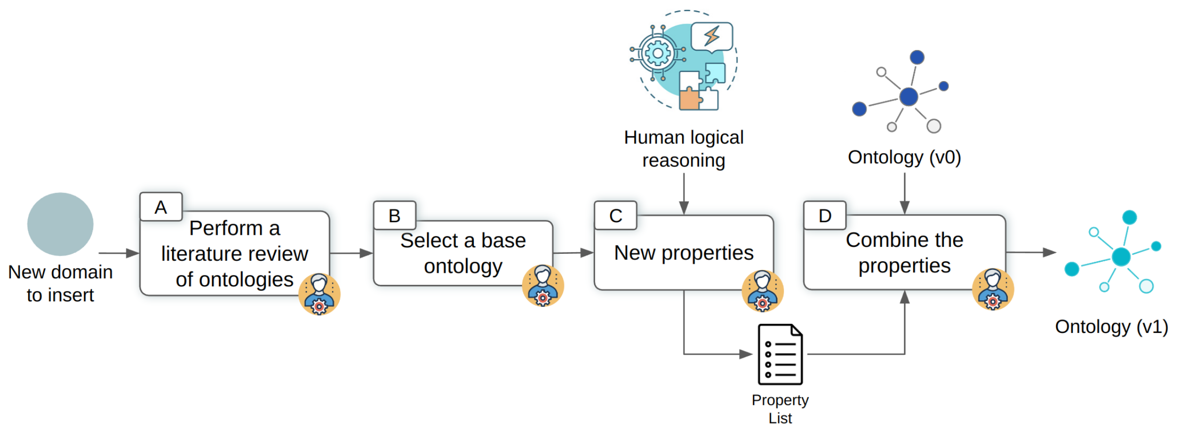


Figure 3.5: Flow of steps to restructure the ontology. (A) Perform a literature review of ontologies: The first step focused on identifying ontologies available in the selected domain; (B) Select a base ontology: The second step focused on selecting the base ontology to support the new domain; (C) New properties. Third step focuses on obtain the new ontological properties needed for the new domain; (D) Combine the properties: The fourth step focused on combining previously detected properties with our ontology `v0`.

[1]https://protege.stanford.edu/

3.4 Populate the KG

The last module aims to insert (or populate) the KG with new triples extracted from a collection of specific questions and answers. These triples have to follow the structure of the new ontology `v1` to be added to the KG. Figure 3.6 presents the flow of steps involved in this fourth module.

This module is based on the research carried out by Sant'Anna *et al.* [29], which addresses the insertion of triplets based on questions and answers. In our proposal, we conceptually adopt the theoretical steps proposed in such research as part of the process of populate a KG. However, it is important to highlight that the application of each of these steps differs significantly from the implementation proposed by [29].

The first step refers to obtaining all the necessary information about the product, such as the GTIN, name, category, SKU, among others. To obtain this information we execute an additional query for each question, this information is retrieved from the database where the product information is stored. A Product refers to an item offered in an *e-commerce.*

The second step consists of extracting the intents and entities from the questions and answers entered. In our solution, this process is based on the work of Sant'Anna *et al.* in a conceptual way [29], who proposed their own Natural Language Understanding (NLU) process, where they defined the concepts of entity and intent.

Sant'Anna *et al.* [29] defines that the entity represents a term or expression with a known meaning relevant to understanding the sentence. In addition to having names and values, for example, the "brand", "model", "voltage", "volume" present in an appliance. The intent is to identify whether the question refers to compatibility between entities. This step is responsible for processing the stored set of questions answered manually by human assistants and product information, a key source of knowledge to update the KG and answer new similar questions. This extraction process is necessary to structure the knowledge through RDF triples. It is relevant to note that [29] uses RASA NLU [8] for the extraction of entities and intents, unlike our proposal that uses GPT-3 LLM to carry out this task.

The third step is the creation of RDF triples, which is a key factor in updating the current KG. This knowledge extraction process is based on extracting intents and entities from each pair of questions and answers (input in our framework) to structure the extracted knowledge in *RDF* triples.

For example, in the question "*Is USB Type-C Charging Cable compatible with the Huawei Mate 20 Lite 6.3 inches ?*" and the answer "*Yes, this product is compatible*"(3.4), we can observe the presence of 3 entities such as *Brand, Model* and *Size.* Additionally, we have information about the product, the intention of the question and finally the type of compatibility present in the answer. In this way we can generate triplets necessary to be inserted later in the KG. Below we show the generated triplets:

- `<USB Type-C Charging, compatible, Huawei Mate 20 Lite 6.3 inches>`
- `<USB Type-C Charging, type, Product>`
- `<Huawei Mate 20 Lite 6.3 inches, brand, Huawei>`
- `<Huawei Mate 20 Lite 6.3 inches, model, P30>`

- `<Huawei Mate 20 Lite 6.3 inches, size, 6.3 inches>`
- `<Huawei Mate 20 Lite 6.3 inches, type, Technology>`

The fourth step is to store the collection of triplets in a triplestore; this can be local or remote. This step aims to access information through SPARQL queries. In this step, one can use software tools that store RDF data, as well as GraphDB[2], Virtuoso[3], among others. The process of storing the triplets can be carried out using a Python script, which is presented in greater detail in the Algorithm 7.

Finally, the output data of this last step is the KG `v1`. This module can be developed automatically because each step can be executed by a script.

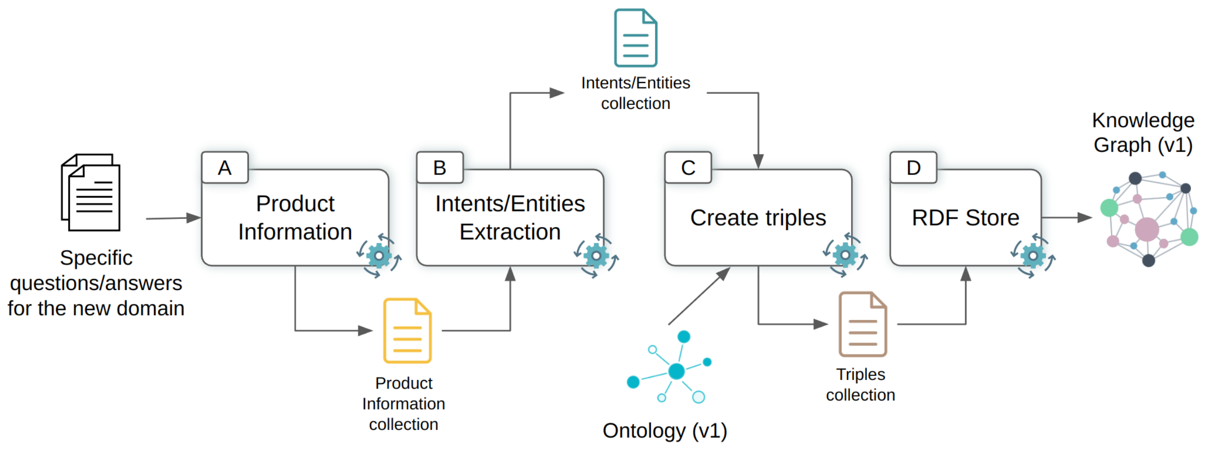

Intents/Entities collection
Knowledge Graph (v1)
Specific questions/answers for the new domain
A
Product Information
B
Intents/Entities Extraction
C
Create triples
D
RDF Store
Product Information collection
Ontology (v1)
Triples collection

Figure 3.6: Flow of steps to populate the KG. (A) Product Information: The first step focused on obtaining the information of each question and answer; (B) Intents/Entities Extraction: Second step focused on extracting the intention and entity of the questions; (C) Creation of triples: Third step focused on create triples to update the KG; (D) RDF Store: Fourth step focused on store the generated triples.

Chapter 4 describes in detail the application of our UpKG framework applied to the context of an AI startup in Latin America.

3.5 Discussion

This Chapter introduces our UpKG framework and describes in detail the insertion of new domains into an existing KG from pairs of questions and answers. The framework consists of four interconnected modules, each with a specific purpose.

The first module focuses on understanding KG, including a thorough analysis of its existing structure, relationships, and ontology, and provides a basis for KG extension. The second module identifies the new domains to be incorporated, which involves identifying unanswered questions in the KG and detecting missing relationships. Once these gaps are identified, the third module restores existing ontology to accommodate new relationships. This process ensures that ontology is consistent and adapts to the new domain you want to

[2]https://www.ontotext.com/products/graphdb/
[3]https://virtuoso.openlinksw.com/

insert. Finally, the fourth module is responsible for populating the KG. For this purpose, it extracts the intentions and entities of the pairs of questions and answers, creating triples inserted in the KG.

This framework presents a well-defined methodology and step-by-step for the expansion of KG to offer a guide for those looking to incorporate new knowledge into their KG, ensuring that the new knowledge is integrated coherently and efficiently. In addition, one of the highlights of our framework is its ability to adapt to various domains with a focus on e-commerce.

Our proposal originates initially for the insertion of new domains in an existing KG. However, UpKG can be applied without problems for the creation of a KG from scratch, considering that not all modules will be applied in their entirety, as is the case of the first module, or some steps of the third module. Therefore, we can say that, despite its initial objective, this framework can be used in various scenarios.

3.6 Final Remarks

This Chapter described our UpKG framework, which aims to insert new domains in KG in the context of e-commerce. Our framework is based on four different modules, and each module is integrated coherently to ensure that the domain insertion process in the KG is efficient. The next Chapter describes the application of our UpKG framework in a real scenario focusing on e-commerce.

Chapter 4

Applying the UpKG Framework in a Case Study

In this Chapter, we describe the application of our UpKG framework in a real scenario, specifically in e-commerce. Our framework was applied in the company GoBots [1], specialized in Artificial Intelligence solutions for e-commerce platforms in Latin America. GoBots offers solutions to structuring compatibility knowledge between products using a KG based on an ontology. In addition, it provides a question-and-answer solution designed to improve the user experience on e-commerce platforms.

This Chapter is organized into three main sections. First, Section 4.1 further presents the case study context and the methods applied in this case. Section 4.2 describes the results obtained from applying our UpKG framework, detailing the results achieved in each framework module. Section 4.3 discusses our research findings and analyzes the results obtained, contributing to a better understanding of our research work.

4.1 Context of Application and Methods

We contextualize where we applied our UpKG framework instantiation. This research was fully carried out within the company GoBots. The main objective of GoBots is to bring Knowledge Graphs and Artificial Intelligence solutions closer to e-commerce systems, allowing more effective interaction with customers.

Currently, GoBots has a KG-based solution that stores information from a specific domain, in this case, “Automobiles". This KG is deployed and in execution. The existing KG was designed to answer client questions related only to this domain. GoBots KG currently answers customer questions through its own service, which receives a question and proceeds to perform two sequential steps as described below:

- Stage one: This stage consists of processing the question by a component called Query Construction, which consists of encoding the question in triplets that the current KG supports, all following the structure of the ontology current 4.2, where the response to the generated query is given by the following attributes:

[1]`https://gobots.ai/`

  - A boolean variable that indicates whether the knowledge was found, indicating the existence of compatibility between the product and the consumer item.
  - The type of compatibility found between the product and the consumer item.
  - The human assistant's complete answer to the question that generated the retrieved knowledge.

  This stage one is performed without human intervention, as it is a completely automatic process for generating questions.

- Stage two: In this stage a human assistant proceeds to answer the question asked, this question with its answer is stored in the GoBots database, in addition the service saves the information of the product that is being asked, such as the ID and product category. Once the assistant's task is completed, the service proceeds to analyze the question. This analysis consists of extracting intentions and entities from the question, then creating triplets based on their ontology as shown in Figure 4.2. Finally, the triples created are stored within a triplestore. The stage two is performed if and only if stage one was not able to answer the question.

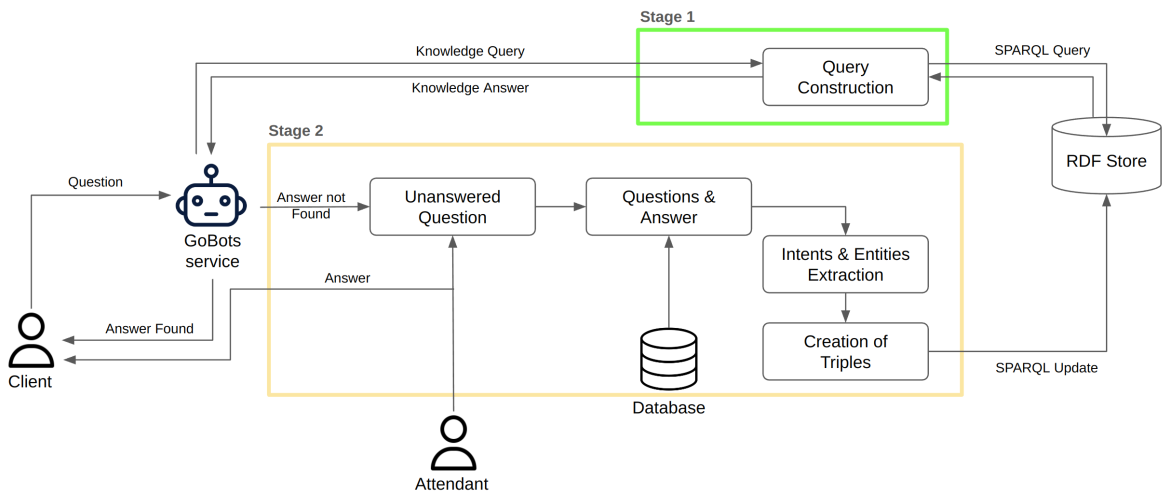


Figure 4.1: Flow used by GoBots to answer customer questions using their KG. The diagram presents two stages, where stage one is the retrieval of information from the KG to answer the question, while stage two is the update of the KG with new knowledge thanks to the response of a human assistant.

Currently the GoBots KG in production consists of 1,430,133 instances of type Compatibility, of which 125,646 are of type NoCompatibility and 1,304,488 are FullCompatibility. The number of Product instances is 269,862 and the number of Car type instances is 25,636. All this information stored within the KG was obtained from the Automobiles domain.

The KG in place uses the ontology `v0` shown in Figure 4.2, where Compatibility is the class that represents the type of compatibility between a Product and a ConsumerItem, it also has two compatibility subclasses, FullCompatibility being the one that represents the existence of compatibility between the ConsumerItem and the Product; while the

NoCompatibility subclass is the opposite, this subclass represents that there is no compatibility between the Product and ConsumerItem classes. The Product refers to the product about which the question is being asked, which generally has a unique identifier in addition to its category. On the other hand, the ConsumerItem class represents the item related to the question. In the current KG we can only observe the presence of the Car class, which represents the Automobiles domain.

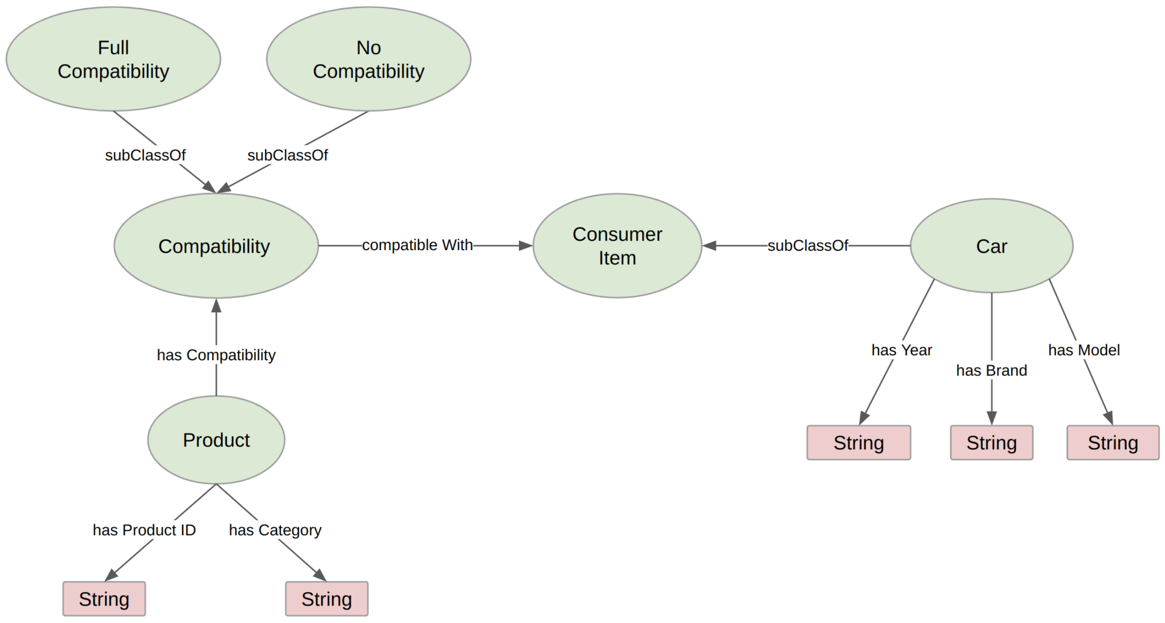


Figure 4.2: Ontology `v0` that represents the compatibility knowledge between a product (a car seatbelt, for instance) and a consumer item (an Audi TT, for instance) [29].

To effectively address potentially more customer questions, we applied our UpKG framework to add a new domain to the KG and to apply UpKG was important the support of the developers of the current KG in GoBots since they are the main experts of the KG and its structure. We also had access to the KG documentation and the GoBots database where they store the questions and answers asked by all the clients. This implementation was carried out within the existing GoBots KG.

With this initiative, we seek to expand the capacity of the KG in place and allow it to answer questions from various domains. This must contribute significantly to improving the user experience and the solution's effectiveness in the e-commerce field.

4.2 Results Applying the UpKG Framework

This section shows the results of applying our UpKG framework in the GoBots company. Section 4.2.1, presents the result of applying the first module, where the understanding of the current KG of Gobots is shown. Section 4.2.2, presents the result of applying the second module which is to identify the new domain to be inserted in the GoBots KG. Section 4.2.3, presents the result of restructuring the ontology, where the process of inserting the properties of the Appliances domain into the Automobiles domain is shown. Section 4.2.4, presents the resulting KG population from the UpKG framework. This section shows the process of how the final KG accepts the Appliances and Automobiles domains.

4.2.1 Results in understanding the current KG

Figure 4.3 presents the application of the first module, which is to understand the current KG where we have as input the KG of GoBots, which we denote as `v0`.

In step A (Figure 4.3), it is required to understand the structure of the KG. In our case, we had access to GoBots documentation and the current KG in production to perform this step. With the documentation, we conceptually understood how the KG was structured, the objective by which the KG was designed, and how this was used to answer questions in an e-commerce environment.

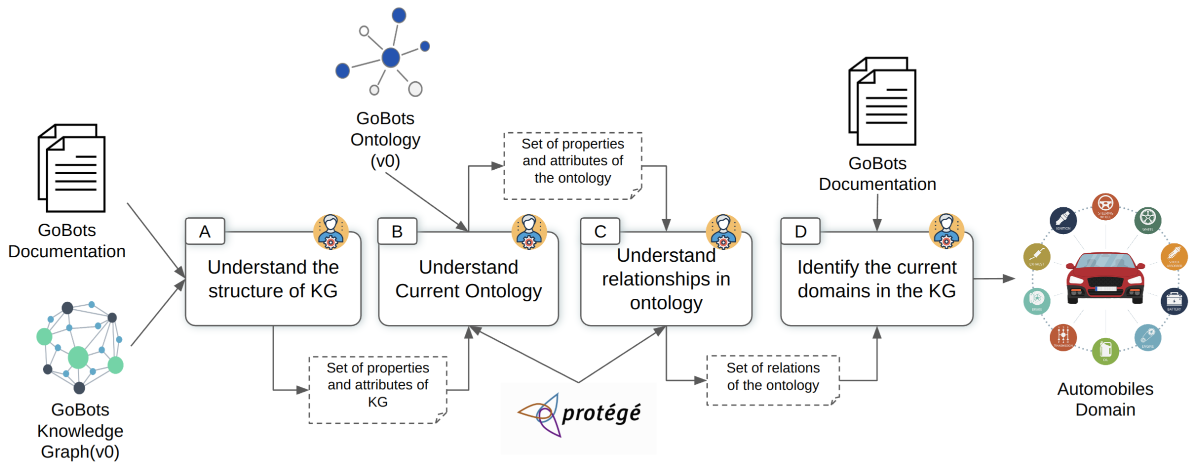

GoBots Ontology (v0)
Set of properties and attributes of the ontology
GoBots Documentation
GoBots Documentation
A
Understand the structure of KG
B
Understand Current Ontology
C
Understand relationships in ontology
D
Identify the current domains in the KG
Automobiles Domain
GoBots Knowledge Graph(v0)
Set of properties and attributes of KG
protégé
Set of relations of the ontology

Figure 4.3: Results in understanding the current KG in GoBots.

As a result of understanding the structure of KG, we have a set of properties and attributes of the KG `v0`, these properties refer to all the existing relationships in the KG `v0`, in the same way we map all the attributes that belong to each class within the KG `v0`. Figure 4.2 shows the initial ontology, in which we observe the existing classes, properties, and attributes to store the compatibility knowledge between a product and a Car type object.

In step B (Figure 4.3), we used the initial ontology `v0` proposed in [29]. In this step, we used the tool Protégé to visualize and understand how the ontology was structured. As a result of understanding current KG, we figure out a set of properties and attributes in the ontology `v0`.

In step C (Figure 4.3), like the previous step, we used the Protégé tool to understand the relationships presented in the initial ontology, which contains the following classes: *Compatibility*(*FullCompatibility* and *NoCompatibility*), *ConsumerItem*, *Product* and *Car*. Where the *Product* class represents a product in an e-commerce system, *ConsumerItem* is an abstract class representing an item that is related to the product; this relationship is given by the *Compatibility* class, either of the *FullCompatibility* and *NoCompatibility* types. As a result of understanding relationships in ontology, we have a set of relations of ontology `v0`, at this moment we have mapped all the existing relationships in the ontology `v0` as well as in the KG `v0`, these relationships for example are: *compatibleWith, hasCompatibility, hasProductID, hasYear, hasBrand, etc.*

In step D (Figure 4.3), we defined the domain(s) that cover the current KG. In this case, the current domain is the *Automobile* only, which was represented by the *Car* class, keeping the following attributes: *hasYear, hasBrand* and *hasModel*, which represent the year, brand, and model of an object *Car*.

4.2.2 Result in identifying new domain

GoBots collected question-answer pairs related to a product that human attendants answered, these question and answer pairs are stored in the GoBots database each time a question is asked to the GoBots service as shown in Figure 4.1. This collection is stored in a MongoDB database. Figure 4.4 shows the process of identifying the new domain to be inserted in the new version of the KG.

In step A (Figure 4.4), we identified all the questions that the current KG `v0` cannot answer. This task was carried out with a Python script that connects directly to the GoBots dataset to extract the question collection within a delimited date range. Additionally, we do not consider the questions related to the domain of Automobiles. In this evaluation, the date range was two months, specifically from March 1, 2022 to May 1, 2022. The output of step A is a collection of questions the KG `v0` did not answer. For this reason, we perform a filter obtaining all the questions related to categories different to Automobiles.

Below we show the Algorithm to extract the total number of questions grouped by their root category (domain). In the GoBots database, each product is associated with a daughter category and not directly with the root category, and each question is associated with a product. Having this information it is possible to obtain the total number of questions asked for each root category. For example, the categories $Air\_and\_Ventilation$ and $Washing\_Machines$ are subcategories of the category *Appliance*, and each subcategory has a number of questions, so this Algorithm shows the steps necessary to group and get all the root categories with their total number of questions.

The Algorithm 1 receives four input parameters described below:

1. Parameter *database*: It is the instance with the necessary access to enter the GoBots database, in this case we will specifically use the collection $pergunte\_aqui$ that contains all the questions asked by clients.

2. Parameter $range\_date$: Range of dates in which the domains are extracted with their associated total number of questions.

3. Parameter *categories*: Set of all the categories present in GoBots, this set has the necessary information to determine the hierarchy between the categories.

4. Parameter *excludes*: Categories not to be considered, in this case it is the root category of Automobiles.

Algorithm 1 Grouping of questions according to their root category.

```
Require: $database$
Require: $range\_date$
Require: $categories$ {All categories hierarchy}
Require: $exclude$ {Automobile category root}
 1: $collection \leftarrow database.pergunte\_aqui$
 2: $pipeline \leftarrow get\_pipeline\_category(range\_date)$
 3: $results \leftarrow collection.aggregate(pipeline)$
 4: $group \leftarrow \emptyset$
 5: for all $result \in results$ do
 6:    $category \leftarrow categories.get(result.id)$
 7:    $root \leftarrow category.root$
 8:    if $root.id \notin exclude$ then
 9:       if $root.id \in group$ then
10:          $group[root.id] \leftarrow group[root.id] + result.count$
11:       else
12:          $group[root.id] \leftarrow result.count$
13:       end if
14:    end if
15: end for
16: return $group$
```

In summary, the Algorithm 1 performs a search for the total number of questions for each root category within a date range and returns which are the domains with the greatest number of questions not answered by the KG `v0`.

In step B (Figure 4.4), we identified manually the relations that are not mapped in the ontology and, therefore, are impossible to answer by the KG. In this step, we analyzed the structure of the questions and compared them with the properties contained in the GoBots KG `v0`. This analysis is carried out manually, and human intervention is necessary to understand the relationships and properties present in the questions from a different domain than automobiles. This analysis is carried out based on recognizing the intentions and entities present in the question as shown. previously in Figure 3.4. In addition, these relationships are necessary to be adapted and inserted into the current ontology, this step refers to the step 4.2.3.

Among them, we have questions like *"Is it compatible with LSE09 220v?"*, where *LSE09* refers to a Washing Machine (Appliance domain) that works with a voltage of *220V*. This example shows that the KG cannot store this knowledge using the *Car* class defined in the ontology `v0`. As a result of step B, we have a set of relationships not mapped by the current ontology.

In step C (Figure 4.4), we created a script in Python to identify the domains (categories) with the largest number of questions not answered by KG `v0` sorted descending; the three domains with the most questions and answers were Appliances, Home Furniture, and Tools.

The Algorithm 2 shows the necessary steps to graphically obtain a histogram of all the root categories, it also returns the category (domain) with the greatest number of unanswered questions, which in our case is *Appliance*. The histogram obtained from applying 2 is the one shown in Figure 4.5. It is important to mention that the process of identifying the new domain is carried out automatically using the Algorithms 1 and 2. To obtain the new domain it is necessary to execute the Algorithms sequentially.

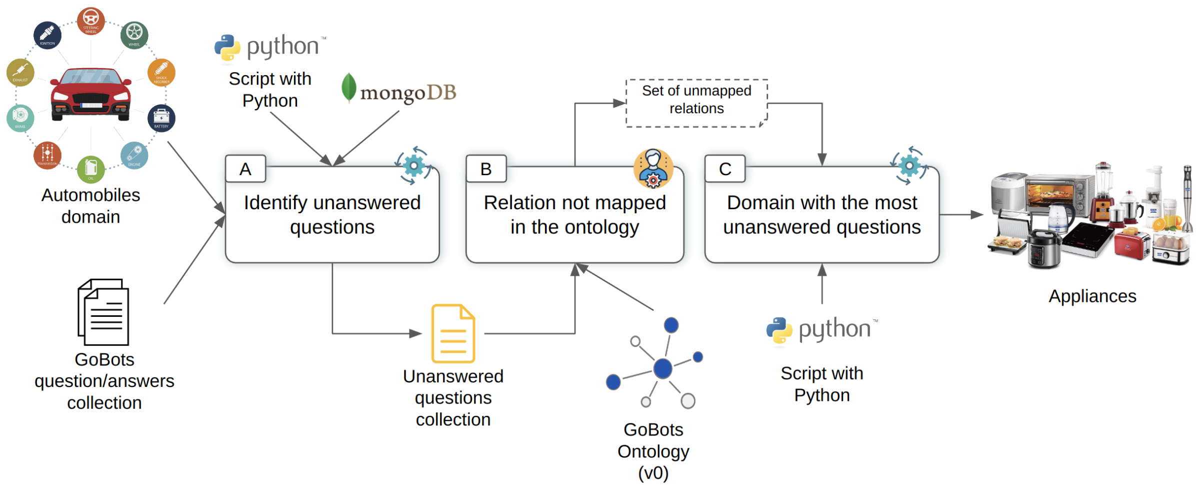

Automobiles domain
GoBots question/answers collection
python
Script with Python
mongoDB
A
Identify unanswered questions
Unanswered questions collection
Set of unmapped relations
B
Relation not mapped in the ontology
GoBots Ontology (v0)
C
Domain with the most unanswered questions
python
Script with Python
Appliances

Figure 4.4: Diagram showing the result of identifying a new domain in GoBots.

Algorithm 2 Histogram of categories based on the number of questions.

```
Require: $group$
1: $ordering \leftarrow sort(group)$
2: $plot\_histogram(ordering)$
3: $first\_domain \leftarrow ordering[0]$
4: return $first\_domain$
```

In the next module, we present the insertion of the Appliances domain to the new version of the ontology and later to KG. Figure 4.5 shows the domain that has the highest number of questions not answered by the KG `v0` is Appliances. For this reason, we inserted this new domain with the intention of covering the greatest number of questions not answered by the KG. In a new application of our framework we take the domain with the highest number of unanswered questions, which is different from Automobiles and Appliances.

4.2.3 Result in restructuring the ontology

The ontology `v0` is limited to the Automobiles domain, which does not allow receiving queries from other domains. That is why when applying our UpKG framework, the need to insert the “Appliances" domain was identified to cover a large part of the questions not answered by the current KG.

In addition, we decided to maintain the *Compatibility* intent between a product and an object from the Automobile domain, this decision was made to maintain the behavior of the initial ontology. The ontology obtained from applying our framework was denoted as `v1`, which can cover two domains of questions users ask in an e-commerce system.

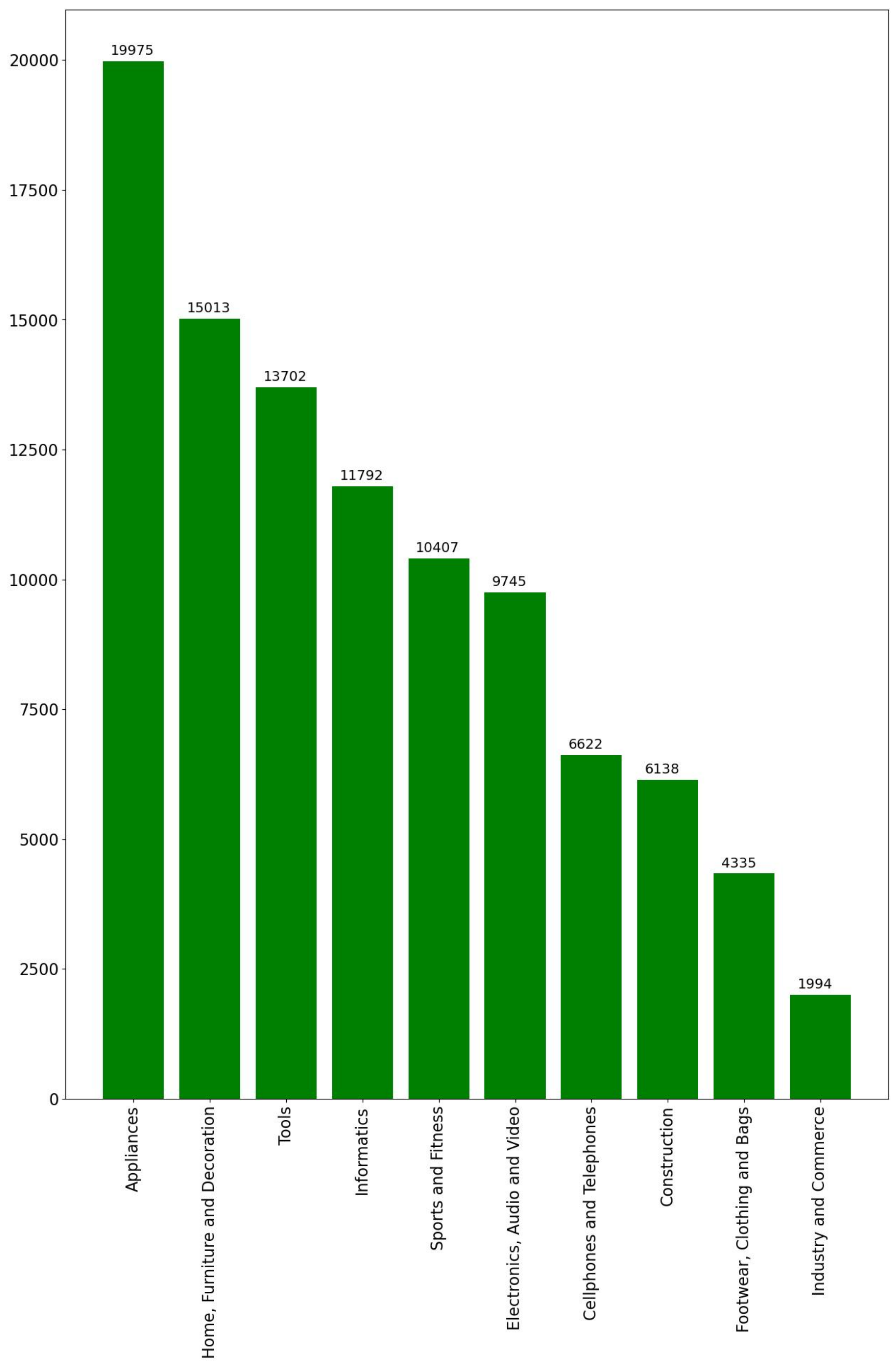


Figure 4.5: Histogram from the root categories with their number of questions.

Figure 4.6 shows the process of restructuring the GoBots ontology to accept the Appliance domain.

In step A (Figure 4.6), we performed an ontology search in literature, specifically in works previously published by the scientific community; in our case, it is to carry out a literary search for ontologies that cover the domain of appliances, or also ontologies focused on products. A search for publicly available product ontologies was carried out because it is intended to expand the KG to several e-commerce categories. We explored *Schema.org*[2], which provides a list of properties in Product and allows us to identify the properties needed for the new domain.

In step B (Figure 4.6), we selected *Schema.org* as the base for our ontology because this source provides a set of standard properties that align perfectly with the domain we want to insert.

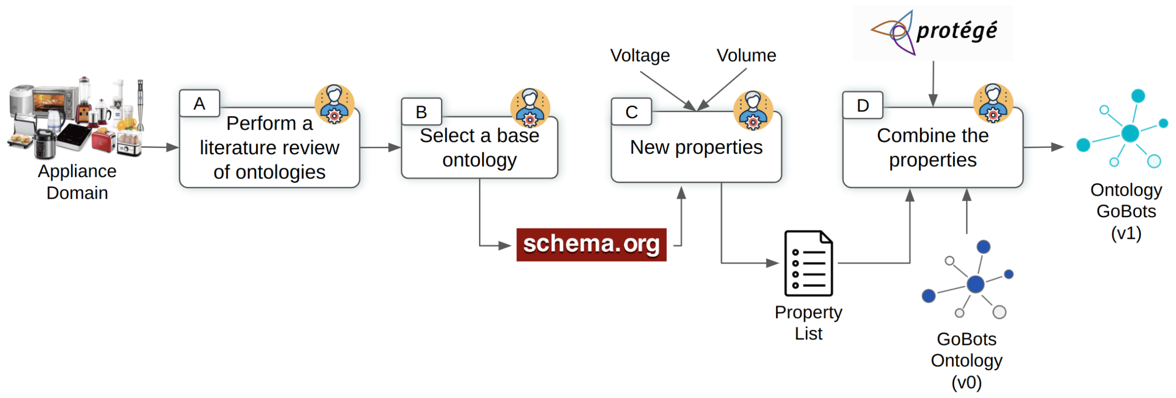


Figure 4.6: Results in restructuring the GoBots ontology with the “Appliance Domain".

In step C (Figure 4.6), we identified the new properties necessary for our ontology, these new properties are identified by human reasoning, taking into account the properties that are necessary to store new knowledge supported by ontology. This analysis is done manually to a collection of questions from the domain of Appliances, as shown in Figure 3.4. In this case, we used standard properties present in *Schema.org* regarding Product, such as Model, Brand, and Material. *Schema.org* presents properties to represent entities that are widely used by different applications, entities such as Products, Events, Health, etc.

We propose new properties that was not found in *Schema.org*. These are proposed from human reasoning to map the relationships in the Appliances domain, that is, the ontology developer's criteria are used to propose properties that are necessary to cover new knowledge of the Appliance domain. These new properties proposed by the developer's criteria are: Voltage and Volume questions.

In step D (Figure 4.6), we combined the properties of the new Appliances domain with the Automobiles domain, this step is carried out completely manually; in this step, we used the *Protégé tool* to combine our property list with the existing GoBots ontology `v0`. Our list of new properties to insert is the result of using the properties present in *Schema.org* that are not within the initial ontology, in this case we manage to represent the knowledge of the Appliances domain adding the property Material, Voltage and Volume. The final result of this step is a new version of the GoBots ontology that accommodates two domains: Automobiles and Appliances. We denoted this ontology as `v1`.

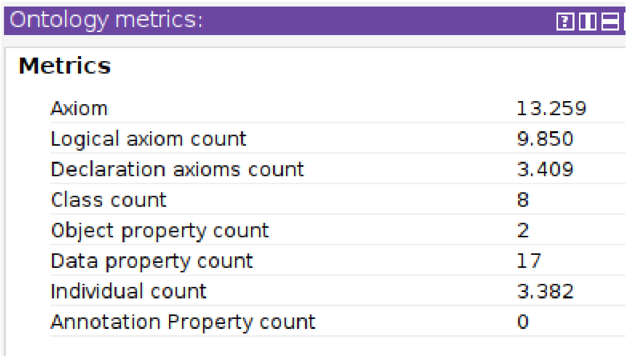
[2]https://schema.org/Product

We added an *Appliance* class that is a subclass of *ConsumerItem* just like the *Car* class to our initial ontology, in which we added all the attributes mapped as: *hasMaterial hasVolume* and *hasVolts*.

Figure 4.10 shows in Protégé tool visualization all the Data Properties that connect to the "Automobiles" and "Appliances" domains, some of which present independent attributes for each domain.

Regarding the Object Properties, our ontology maintains the two existing relationships between the *Product* and the *ConsumerItem* through a Compatibility type proposed in [29]. These are represented by *compatibleWith* and *hasCompatibility*, where *compatibleWith* ranges to a *ConsumerItem* and domain to a *Compatibility* type. However, *hasCompatibility* has a *Compatibility* type as range and a *Product* domain.

Figure 4.7 shows the metrics of the ontology structure in terms of classes, properties, and axioms. The ontology `v0` presented seven classes, while our ontology `v1` presented eight classes. This shows that inserting a new domain in ontology was not complex regarding new classes added, considering that ontology `v0` was already well-designed to accept other domains.


Ontology metrics:

Metrics

Axiom	13.259
Logical axiom count	9.850
Declaration axioms count	3.409
Class count	8
Object property count	2
Data property count	17
Individual count	3.382
Annotation Property count	0

Figure 4.7: Metrics of the ontology `v1`

Figure 4.8 presents how the ontology `v1` was structured, which was obtained at the end of the ontology restructuring phase as explained in Section 4.2.3. This new ontology `v1` consists of a new class *Appliance* that is a subclass of *ConsumerItem*, where the class *Appliance* is responsible for storing the new knowledge extracted from the questions asked by users. In addition, the compatibility relationships between the classes remain the same, thanks to the advantage that *ConsumerItem* is the class with which the *Compatibility* class is related. This ontology was designed as an extension of the ontology proposed in [29]. This choice was made in order to simplify the migration from the previous KG to the new KG, thus avoiding significantly compromising the initial structure.

Figure 4.9 presents the Object Property of the ontology `v1`. The *compatibleWith* property represents the relationship that exists between the class *Compatibility* and *ConsumerItem*, where the compatibility type is represented as *FullCompatibility* or *NoCompatibility*. Additionally, we have the property *hasCompatibility*, which indicates whether a product has some type of compatibility. Having these two Object Properties we can make queries between *Product* and *ConsumerItem*. This relationship is important for answering compatibility questions.

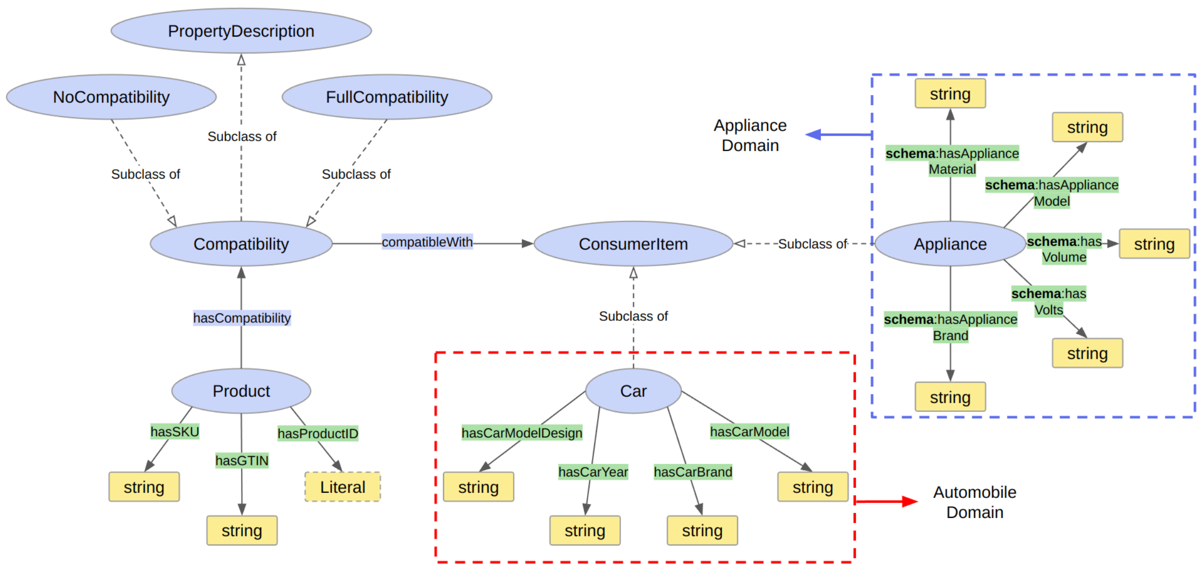
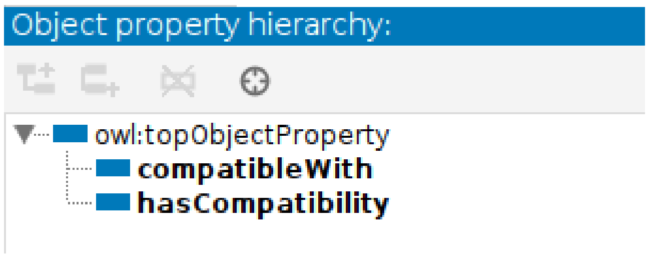


Figure 4.8: Ontology structure `v1`. This represents the compatibility between products and an object of Car or Appliance type.

Importantly, the existing KG of *GoBots* was developed for the purpose of representing the compatibility relationship between a buyer's product (*ConsumerItem*) and an e-commerce product (*Product*), thus being easier to represent compatibility between two different classes even though both represent a product. For this reason, we have kept these two classes for each type of product in our proposal.


Object property hierarchy:

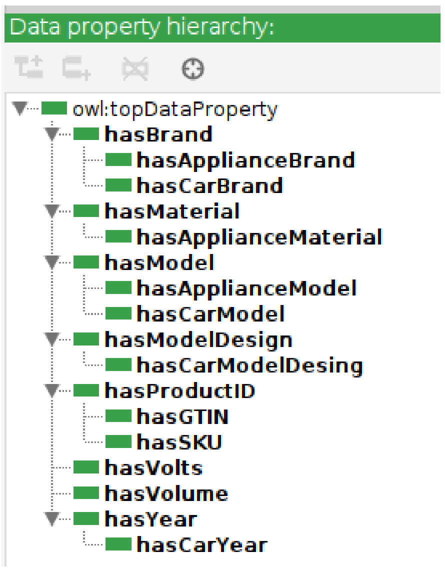
owl:topObjectProperty
compatibleWith
hasCompatibility

Figure 4.9: Object Property of the ontology `v1`

Figure 4.10 shows Data Property of the ontology `v1`. As the classes *Car* and *Appliance* share attributes such as *Brand* and *Model*, we consider creating sub-properties for the different domains and thus maintain the properties directly with the *ConsumerItem*. This way of representing knowledge allows us to easily update the ontology without losing the relationships already built with the *DataProperty*

4.2.4 Result in populating the KG

To populate our KG, we collected a set of client questions and attendant answers from 10 merchants from the GoBots database between September 18, 2023 and October 02, 2023, obtaining a total of 7557 questions. We automatically filtered out those questions with compatibility intent, because the real purpose of the KG is to answer questions that have


Data property hierarchy:

```
owl:topDataProperty
    hasBrand
        hasApplianceBrand
        hasCarBrand
    hasMaterial
        hasApplianceMaterial
    hasModel
        hasApplianceModel
        hasCarModel
    hasModelDesign
        hasCarModelDesing
    hasProductID
        hasGTIN
        hasSKU
    hasVolts
    hasVolume
    hasYear
        hasCarYear
```

Figure 4.10: Data Property of the ontology `v1`

the intention of compatibility, obtaining 1,338 questions. A merchant within GoBots is a seller who owns multiple stores, and each store has its own unique ID.

The questions and answers pairs are obtained from the GoBots database following the following Algorithm 3, specifically from merchants that have stores within Brazil, for which these questions were asked and answered in their original language, in Portuguese. These questions and answers pairs are mostly short, as shown in the following examples:

- Q1: Boa noite. Serve no Eletrolux Eco turbo?.
- Q2: Boa tarde Serve panela oster modelo 8030.
- Q3: Boa tarde. Serve na Frontier Attack 2019?
- Q4: Serve na geladeira CRA3408ANA CONSUL 340 LITROS 110V?

The Algorithm 3 shows how the questions and answers necessary in this module are obtained to be processed and inserted in the new KG. The Algorithm 3 receives 4 input parameters, which are described below:

1. Parameter *database*: This parameter represents the connection to the GoBots database.
2. Parameter *merchant_id*: Id of the merchant, necessary to extract all the questions that were asked to the merchant.
3. Parameter *range_date*: Date range in which the questions were asked.
4. Parameter *category*: All questions are obtained with the domain in the module 4.4, in this case it is the domain *Appliance*.

The main task of the Algorithm 3 is to obtain all the questions and answers of the selected domain (Appliances). Line 1, refers to the collection where the questions asked

Algorithm 3 Extracting questions and answers related to a merchant

```
Require: $database$
Require: $merchant\_id$
Require: $range\_date$
Require: $category$ {Domain (Automobiles or Appliances)}
1: $collection \leftarrow database.question$
2: $store\_ids \leftarrow get\_stores\_by\_merchant(database, merchant\_id)$
3: $category\_ids \leftarrow get\_child\_categories(category)$
4:
5: $pipeline\_question \leftarrow pipeline\_question(range\_date, category\_ids, store\_ids)$
6: $pipeline\_attributes \leftarrow pipeline\_attribute()$
7:
8: $questions\_answer \leftarrow collection.find(pipeline\_question, pipeline\_attributes)$
9: return $questions\_answer$
```

by users are stored. Line 2, we have the variable $store_ids$ that represents the ids of all the stores that the merchant has. A merchant has at least one store available. The function $get\_stores\_by\_merchant()$ makes a connection to the GoBots database to obtain all the ids of the stores that the merchant has. Line 3, we obtain all the ids of the subcategories of the Appliance domain. Line 5, refers to the function that obtains the filter of the questions based on the date range, subcategories and stores, while Line 6 represents the attributes that we want to obtain from the collection of questions and answers. Finally, Line 8 is where we execute the query based on the two previously defined pipelines.

Figure 4.11 presents the automatic process of populating the KG.

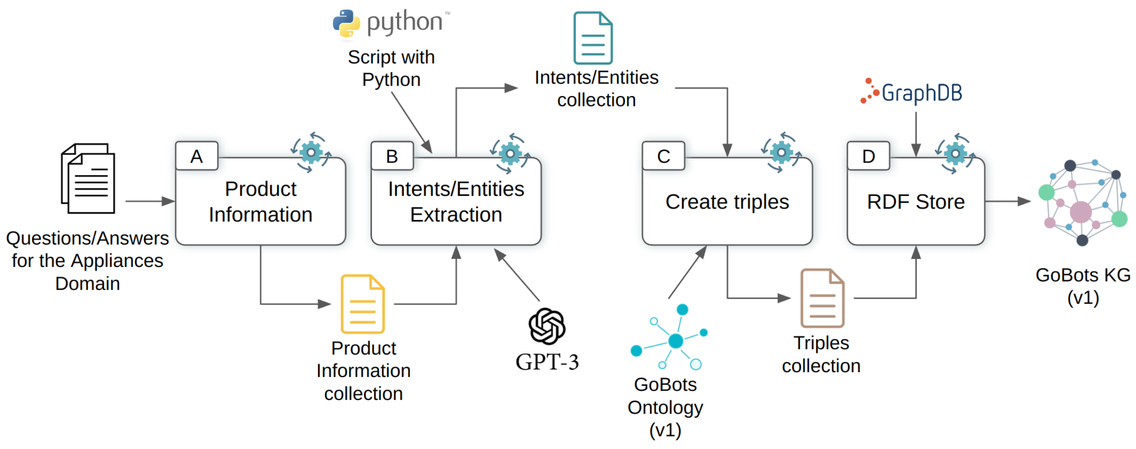


Figure 4.11: Results in populating the KG with Appliance Domain.

In step A (Figure 4.11), we obtained the product information related to the question. This extraction of information was carried out by a POST request, which returns the necessary attributes such as SKU, GTIN or ProductID, which are unique identifiers of a product. This information is required to create a product in our ontology `v1` (cf. Figure 4.8). The Algorithm 4 shows the sequence of steps necessary to obtain essential information for each question and answer. Until now we have all the questions related to the Appliances domain along with its product information.

Algorithm 4 Extraction of product information

```
Require: $service\_url$
Require: $questions$
 1: for all $question \in questions$ do
 2:    $url \leftarrow service\_url + question.productID$
 3:    $response \leftarrow request.get(url)$
 4:    for all $attribute \in question.attributes$ do
 5:       if $attribute.name == GTIN$ then
 6:          $questions.gtin = attribute.value$
 7:       else if $attribute.name == SKU$ then
 8:          $questions.sku = attribute.value$
 9:       end if
10:    end for
11: end for
12: return $questions$
```

In step B (Figure 4.11), we extracted the intents and entities of the questions and answers collected previously in step A. This collection of questions and answers is related to the domain of Appliances and Automobiles. In this step, we defined and created a solution via a Python Script to automatically identify the questions with the "Compatibility" intent type. This script is represented by the following Algorithm 5, which consists of filtering only the questions that have the intent of compatibility, in addition to guarantee that the question is compatible we use a reliability minimum of 80%.

Algorithm 5 Filter the compatibility questions

```
Require: $service\_url$
Require: $questions$
 1: $questions\_compatibility \leftarrow \emptyset$
 2: for all $question \in questions$ do
 3:    $payload \leftarrow question.text$
 4:    $response \leftarrow request.post(service\_url, payload)$
 5:    $intents \leftarrow response.intents$
 6:    if $intents[0].name ==$ "Compatibility" and $intents[0].confidence > 0.8$ then
 7:       $questions\_compatibility.append(question)$
 8:    end if
 9: end for
10: return $questions\_compatibility$
```

We obtained 7557 questions and answer pairs from ten merchants, distributed equally between five appliance domain merchants and five automobile domain merchants. Once the questions were filtered with the compatibility intention, we obtained 1338 questions and answer pairs, of which 241 were related to the Appliances domain and 1097 to the Automobiles domain.

After we proceed with the automatic extraction of the entities involved, for example, the question *"Is it compatible with LSE09 220v?"*, from which we extract the entities in place, such as the "*Model*" and "*Voltage*". To perform this extraction, we developed a script that receives a natural language question as input and returns all identified entities (domain, model, brand, volume, volts, design, and material).

At this stage, our solution used the OpenAI[9] company API [3] to extract **entities** from the question collection via the use of the GPT-3 large language model (GPT-3 LLM) and the use of a prompt. We use GPT-3 LLM as a model to extract entities within a question, this is because entity extraction is a complex task taking into account that the application is carried out in different domains. However, GPT-3 LLM can be replaced by some other method that implies greater reliability in the extracted entities. In this solution, we defined the function `Prompt`, which aims to send text to the model to request a response; that is, the function `Prompt` acts as the initial indication for the model, and the response generated by the model is based on the content and context of that `Prompt`, in our case we provide the context in which we are making a query to the API GPT-3 LLM, as seen in Table 4.1. The `Prompt` is an important part of the interaction with the model via API because it defines the query you aim apply to the model. Defining a clear and specific `Prompt` for accurate and relevant answers is important.

The following Algorithm 6 represents all the steps performed for the extraction of the entities within the question, as well as the answer to the question that was given by a human assistant. This Algorithm has 2 input parameters, the first *questions_compatibility* refers to all questions that have the intention of compatibility, the second parameter $prompt_type$ is a variable that indicates to which domain the question belongs.

Algorithm 6 Extract the entities from the question

```
Require: $questions\_compatibility$
Require: $prompt\_type$
 1: $questions\_entities \leftarrow \emptyset$
 2: for all $qa \in questions\_compatibility$ do
 3:     $prompt \leftarrow get\_prompt(prompt\_type)$
 4:     $new\_prompt \leftarrow prompt + str\_prompt(prompt\_type, qa.question, qa.answer)$
 5:     $response \leftarrow query\_to\_model\_openai(new\_prompt)$
 6:     if $validate(response)$ then
 7:         $questions\_entities.append(response)$
 8:     end if
 9: end for
10: return $questions\_entities$
```

The `Prompt` defined in our solution is as shown in Table 4.1, this is used through a Python script and aims to identify and structure questions related to products and extract information relevant to the questions. It should be noted that we use two `Prompts`, one of them is for the Automobiles domain and the second for the Appliances domain. The information to be extracted includes the question category, make, model and year for questions related to the Automobiles domain; while for the Appliances domain we use the second Prompt, which has the function of extracting the category of the question, the model, the brand, the voltage and the weight of the appliance.

In addition, three examples of appliance-related questions were described in the prompt,

[3] `https://platform.openai.com/docs/api-reference`

Table 4.1: Prompts used for extracting entities within a question.

Domain	Prompt	Example
Automobile	You have to analyze the question and extract the category from the question. In the question field, if it is automotive, you must extract the entities, extract the automotive brand, extract the automotive model, extract the year from the automotive. Also, interpret the response and extract whether the response is of type “Compatible" or “Not Compatible". I will give you 3 examples and I want you to return the fourth example following the indicated format. { Question: Is it compatible with Chevrolet Agile 2012, Answer: Good morning, yes it works, Intent: Compatible, Category: Automotive, Brand: chevrolet, Template: Agile, Year: 2012 }	Question: Is it compatible with Gol 16v Turbo 2001? Answer: Good afternoon! this is not compatible Intent: Not Compatible Category: Automotive Brand: Gol Model:Turbo Year: 2001
Appliance	You have to analyze the question and extract the category of the question. In the case of an appliance you must extract its entities, extract the brand of the appliance, extract the model of the appliance, extract the voltage of the appliance, extract the weight of the appliance. Also, interpret the response and extract whether the response is of type “Compatible" or “Not Compatible". I will give you 3 examples and you must return the fourth example following the indicated format. { Question: Good morning, could you tell me if this model is compatible? with the Consul CWB09BB 9KGS B 110V washer, Answer: Good afternoon! this is not compatible, Intent: Not Compatible, Category: Appliance, Brand: Consul, Model: CWB09BB, Voltage: 110v, Weight: 9KGS }	Question: Is this board compatible with the Brastemp washing machine model BWB11ABANA30 Answer: Good morning, yes it works Intent: Compatible Category: Appliance Brand: brastemp Model: BWB11ABANA30

along with the details expected to be extracted from the questions, these details referring to the entities present in the Appliance domain. In the same sense, we performed a `Prompt` for the domain of Automobiles.

Extracting information via LLMs might return hallucinating responses and interfere with correct information processing [18]. These hallucinations can change the original question, as well as define the question as another category other than “Appliances" or “Automobiles". To combat this difficulty, we carry out a double check validating if the question that the LLM is processing is the same one that was asked, this to prevent the original question from being changed; If it is different from the original, we discard the entire response generated by said LLM model. On the other hand, we compare the category of the original question (the domain) with the category extracted by the model, if they are different we proceed to remove these questions, in this way we ensure that all the questions processed are from the domain to insert.

As a result of this extraction, we obtained a collection of questions with compatibility intent and the entities extracted from those questions.

In general, the Algorithm 6 is responsible for extracting the entities within the question, Line one, represents an empty list that will store the entities extracted by GPT-3 LLM. Line two, we perform an iteration through all the compatibility questions, each iteration is responsible for obtaining the `prompt` designed depending on the domain of the question 4.1. Line four, we add a new question within the `prompt` that does not have its entities, this new record will be completed with the response from the OpenAI GPT-3 LLM API following the structure of the designed `prompt`. Line six, we perform a validation to avoid inconsistencies and hallucinations by GPT-3 LLM. If the validations are passed satisfactorily, we proceed with the storage of the entities for each question. Once all the iterations are finished, we return the *questions_response* list of all the questions with their entities and compatibility type.

In step C (Figure 4.11), we created the RDF triples; it is necessary to emphasize that this step was performed automatically, like the previous two. The creation of triples follows the flow developed in the Algorithm 7. This step generated all the relationships allowed by our ontology `v1` for the different domains, generating instances of “Automobiles" and “Appliances" by triples.

The Algorithm 7 is responsible for creating the triplets from the entities found in the previous step. That is why we perform a loop to generate the triplets of all the questions that have mapped entities. X represents the instance of *ConsumerItem*, in this case it can be from the Automobiles or Appliances domain. In line 3-5 we have the creation of all the triplets with respect to the previously extracted entities. Finally we have the creation of instances of the Product, Compatibility type and their relationship through *hasCompatibility* and *compatibleWith*.

In step D (Figure 4.11), we stored the data in the RDF Stores. In this step, for implementation purposes, we explored GraphDB[4], a database that stores RDF graphs. This tool was used to store the collected triples and execute SPARQL queries, thus populating the new KG `v1` with questions and answers from the Appliances and Automobiles domain.

To populate our KG, we processed the questions of ten merchants: five are domain

[4]https://www.ontotext.com/products/graphdb/

Algorithm 7 Generation of RDF triples

```
Require: $questions\_compatibility$
Require: $questions\_entities$
Require: $new\_kg$
Require: $domain$
 1: for all $compatibility, entities \in (questions\_compatibility, questions\_entities)$ do
 2:     $X \leftarrow instance(ConsumerItem(domain, entities))$
 3:     for all $entity \in entities$ do
 4:         $new\_kg.newTriples(X, entity.name, entity.value)$
 5:     end for
 6:     $P \leftarrow instance(Product(compatibility.product))$
 7:     $C \leftarrow instance(Compatibility(entity.intent))$
 8:     $new\_kg.newTriple(P, hasCompatibility, C)$
 9:     $new\_kg.newTriple(C, compatibleWith, X)$
10: end for
11: return $questions\_response$
```

Appliances, and five are domain Automobiles. The Automobile domain was also explored together with the Appliance domain to guarantee that the entity extraction process can be carried out by the same approach, which is to make use of LLM models, as is the case of GPT-3 LLM. Table 4.2 shows the number of questions and answers processed for each merchant in the appliance domain. Please note that the questions and answers collected from the GoBots database were from September 18, 2023 to October 02, 2023. This table consists of the following columns:

(i) **N° of stores**: This column represents the number of stores available for each merchant.

(ii) **Total QA**: This column shows the number of question and answer pairs collected from the GoBots database.

(iii) **Compatibility**: The questions in the previous column were filtered based on compatibility intent type, and those with a value greater than 0.8 confidence have been selected.

(iv) **Error**: This column shows the number of questions that presented problems, either because the LLM explored could not process them or due to lack of information in the input question.

(v) **QA to insert**: This column shows the total number of questions to be inserted into the new KG, and is the result of the column difference between the column Compatibility(iii) and Error(iv).

(vi) **QA without type**: This column performs a filter to remove questions with hallucination generated by the LLM API query, eliminating questions with categories other than “Appliances" and “Automobiles" to clean the information and process the right questions. This filtering is done based on the Algorithm 6, we can eliminate the questions and answers that have hallucinations taking into account the answer obtained by GPT-3 LLM using the `prompt` developed, since we have answers where the question that returns the model is totally different from the original question, we also apply a double verification of the category of the question, since in some cases the model GPT-3 LLM gives us categories that are totally different from what they should be.

(vii) **QA for KG** `v1`: This column shows the number of questions to be entered into KG `v1` and these values are the results of the difference between QA to insert(v) and QA without type(vi).

Table 4.2 presents results for five merchants processed in the "Appliances" domain, with 2753 questions and answers pairs extracted. These pairs were then filtered by compatibility intent type and had a confidence value greater than 0.8, it is important to note that we kept the acceptable minimum proposed by [29], obtaining 481 pairs. Finally, the QA pairs that presented errors during the extraction of entities and intents and did not have a type were removed so as not to affect the KG's quality, obtaining 241 QA pairs to be inserted into the KG `v1`.

Table 4.2: Table of questions and answers pairs for the "Appliances" domain from 5 merchants. Where the second column represents the number of stores of each merchant, the third column the number of questions and answers collected, the fourth column the number of questions that have the intent of compatibility with confidence greater than 0.8, the fifth column the number of questions that could not be processed correctly and that present errors, the sixth column the number of questions processed to be inserted, the seventh the number of questions without type or category *Appliances* and finally the eighth column represents the final number of questions to be inserted to the new KG `v1`.

	N° of stores	**Total QA**	**Compat. (>0.8)**	**Error**	**QA to insert**	**QA w/o type**	**QA for KG** `v1`
Merchant 1	1	1205	90	52	38	4	**34**
Merchant 2	2	471	59	24	35	7	**28**
Merchant 3	3	121	23	4	19	1	**18**
Merchant 4	2	248	83	11	72	13	**59**
Merchant 5	1	708	226	99	127	25	**102**
Total	9	2753	481	190	291	50	**241**

Table 4.3 shows the results for the number of questions and answer pairs processed for each merchant in the "Automobiles" domain. Table 4.3 was processed in the same date range as Table 4.2 and presents the same columns. Table 4.3 presents five merchants processed in the Automobiles domain, with 4804 questions and answer pairs extracted. These pairs were then filtered by compatibility intent type and have a confidence value greater than 0.8, obtaining 1452 pairs. Finally, the QA pairs that presented errors during the intents or entity extraction, and did not have a type were removed so as not to affect the KG's quality, obtaining 1097 QA pairs to be inserted into the same KG `v1`. Common

errors refer to questions that the GPT LLM model is not able to extract the entities, another error is that the category of the original question does not match the category obtained through GPT-LLM.

Table 4.3: Questions and answers pairs for the "Automobiles" domain from five merchants. Where the second column represents the number of stores of each merchant, the third column the number of questions and answers collected, the fourth column the number of questions that have the intent of compatibility with confidence greater than 0.8, the fifth column the number of questions that could not be processed correctly and that present errors, the sixth column the number of questions processed to be inserted, the seventh the number of questions without type or category *Automobiles* and finally the eighth column represents the final number of questions to be inserted to the new KG `v1`.

	N° of stores	**Total QA**	**Compat. (>0.8)**	**Error**	**QA to insert**	**QA w/o type**	**QA for KG v1**
Merchant 1	4	184	41	7	34	2	**32**
Merchant 2	1	437	71	35	36	5	**31**
Merchant 3	2	229	89	46	43	6	**37**
Merchant 4	5	3677	1146	160	986	62	**924**
Merchant 5	1	277	105	23	82	9	**73**
Total	13	4804	1452	271	1181	84	**1097**

At the final stage, the KG `v1` was populated with 1338 questions and answer pairs from the Automobiles and Appliances domain, totaling 3382 instances. This KG `v1` results from applying our UpKG framework, allowing the "Appliances" domain to be integrated into the "Automobiles" domain.

Table 4.4 shows the compatibility type in each domain of KG `v1`. There are 856 questions and answers pairs of type "Full Compatibility" and 482 of type "No Compatibility". Automobiles domain has the largest number of compatibility type questions, and in this domain a greater number of instances can be obtained thanks to the answers given by the assistants, which does not happen with the new domain of Appliances. However, the application of our framework shows the viability of extending the domain to others, and consequently provides greater robustness and knowledge in the resulting KG.

Table 4.4: Compatibility type in the “Appliance" and “Automobile" domain. The second and third columns show the number of questions and answers that are compatible and not compatible between a *Product* and a *ConsumerItem*. A total of 1338 Q&A pairs were inserted into their particular domains.

	Full Compatibility	**No Compatibility**	**Total**
Appliances Domain	136	105	**241**
Automobiles Domain	720	377	**1097**
Total	**856**	**482**	**1338**

4.3 Discussion

This Chapter described the application of our UpKG framework in the e-commerce context, where each module of our framework was applied based on the context, data and documentation provided by GoBots company.

The application of our framework demonstrated that it is possible to insert new domains in a KG, achieving an expansion of knowledge. However, this domain expansion using our UpKG framework brings with it the investment of a reasonable time to carry out the entire framework, since we have two modules that need human intervention, necessary and mandatory time to understand and improve the structure of knowledge. On the other hand, we have that the continuous application of our UpKG framework would lead to a reduction in the execution time of the same. This is relevant because it facilitates the adaptation and updating of the KG to the new needs of a constantly changing market and improves the response capacity in customer service. Each module of our framework has a main objective and has its results.

Each module comprising our UpKG framework has its main objective and challenges. Module 4, which is to populate the KG, was the one that presented the most challenges because automatically identifying intentions and entities from users' questions and answers pairs refers to a complex and difficult task.

Our study explored the use of LLMs for this purpose. This tool successfully allowed us to extract entities from the collection of questions and answers. However, some questions and answer pairs returned hallucinations, which interfered with the integrity of the information. This problem did not affect the population stage of our KG `v1`, since the questions that were altered and different from the accepted domains were removed, in this case Automobiles and Appliances.

As a result of applying our framework, we achieved a KG `v1` further populated with more 1338 questions and answer pairs from the “Automobile" and “Appliance" domains.

As next steps we have the continuous and constant improvement of the automated modules, in the same way looking for and proposing alternatives that allow us to reduce the time of application of our framework, either in improving automated processes such as the automation of processes that are still manual. On the other hand, we consider that the execution of our framework is replicable by other people than the author without presenting difficulties, since we try to make the modules as simple as possible. Additionally, while our framework arises within the context of e-commerce, UpKG can be used in other contexts outside of GoBots, since the four main modules of our framework have conceptual bases, and its implementation is adaptable to other contexts.

4.4 Final Remarks

This Chapter detailed the implementation and application of our UpKG framework in GoBots context, we also show how the GoBots KG currently works and how this is used to answer questions from the Automobile domain. This Chapter provided a detailed description of the application of each of its all modules end to end in the process. The entire process, from the detection of the new domain, the restructuring of the ontology of GoBots, and the insertion of the "Appliance" domain in the KG. We provided detailed information on the number of questions and answer pairs newly inserted in the new version of the KG, this new inserted knowledge was extracted from questions and answers of the two domains that currently covers the KG `v1`. The following Chapter aims to evaluate both ontology and KG outcome, which are the final products of the application of our framework.

Chapter 5

Evaluating the Outcomes from the UpKG Framework

To ensure the viability and applicability of our UpKG framework, conducting a series of evaluations is mandatory. These evaluations are critically important, as they are designed to ensure that the outcome of UpKG implementation can be materialized effectively and efficiently. In this context, it is important to evaluate the result itself, *i.e.*, the generated new ontology and KG. These two artifacts are essential outputs from the framework execution, and consistency and coherence must be rigorously validated. In addition, evaluating the output of our UpKG framework is important to measure its consistency, and the quality of the response returned from its use.

This Chapter presents two distinct approaches to evaluation: (i) Evaluation of the ontology, which aims to ensure consistency and coherence of this artifact, and (ii) Evaluation of the Knowledge Graph, which aims to measure the quality of the triples encoded in the KG considering the accuracy of the responses generated by it. We clarify that this evaluation relies on the generated results (ontology and KG) achieved from Chapter 4 (applying the framework in the GoBots company context).

In the first approach, we propose UpOntology which is a new evaluation approach and allows evaluating ontology through iterations. UpOntology aims to obtain a continuous improvement of the ontology in each iteration. Each iteration is composed of a series of evaluation methods, and the result of each of these evaluation methods determines whether ontology needs improvement. If problems are identified, the improvement phase begins and its objective is to solve the problems identified when executing the evaluation methods. During this improvement phase, the errors identified in the under-evaluation ontology are addressed and resolved, which leads to a substantial improvement in the quality and coherence of the ontology. If we do not identify problems in the execution of each evaluation method, we can affirm that our ontology is consistent and coherent. This iterative process is performed by the ontology developer, who repeats such evaluation methods to ensure that the resulting ontology meets established high quality standards and requirements. This iterative approach is based on constant and continuous ontology improvement.

Note that this ontology evaluation approach is applied after the Third Module (cf. Section3.3) of the UpKG framework. The result of this ontology evaluation approach allows continuing with the Fourth Module 3.4, because ensuring that our ontology is coherent and consistent allows to populate the KG correctly, representing knowledge in the best possible way. Therefore, the Fourth Module 3.4 is applied if the evaluation of the ontology is coherent and consistent.

The evaluation methods used to evaluate our ontology are Reasoner Pellet, OOPS! and OntoDebug. We decided to apply these three validation methods because these methods contribute to a more complete and accurate validation of ontology. Reasoner Pellet is an inference engine, OntoDebug is a plugin that allows to identify inconsistent axioms of greater complexity, and OOPS! provides a detailed list of common ontology pitfalls.

The application of our approach UpOntology has as main objective to evaluate the quality of our ontology developed, where the input data in this approach is the first version of our ontology, which is naturally the result of the third step (3.3), this ontology before applying UpOntology is shown in Figure 5.5. The individual evaluation methods were applied sequentially as it was applied only by the ontology developer. However, this approach can be applied in parallel by different developers or experts, since each method has no dependence on the other methods. The improvement process within UpOntology, it is completely manual, where we make all the necessary improvements based on the errors found by each evaluation method.

In the KG evaluation approach, we propose to evaluate via Competency Questions (CQs), which allows us to guarantee the quality of the data encoded in the KG from an application perspective. Evaluating a KG with CQs allows us to verify if the KG can provide accurate and consistent answers to key questions related to a certain context. This ensures that the final KG of our framework meets its purpose of answering compatibility questions in some domain or other. This evaluation approach was applied after the Fourth Module of our framework (cf. Section 3.4), which corresponds to the KG population process.

5.1 Evaluation Approaches

To evaluate the ontology and the KG generated through the application of the UpKG framework, we proposed two evaluation approaches: (i) Ontology Evaluation; and (ii) KG Evaluation. Figure 5.1 presents these two assessment approaches. It is essential to highlight the context in which these evaluations were carried out. In the case of ontology evaluation, the third module of our framework was completed. On the other hand, the KG evaluation was performed at the end of the execution of all modules of the UpKG framework (in our case study). This distinction is important, as it highlights the timing of each component and gives us a clear view of the evaluation process in developing and applying our approach.

We highlight that the decision was made to evaluate ontology and KG separately because creating ontology is a preliminary and fundamental step to continue knowledge insertion in the KG.

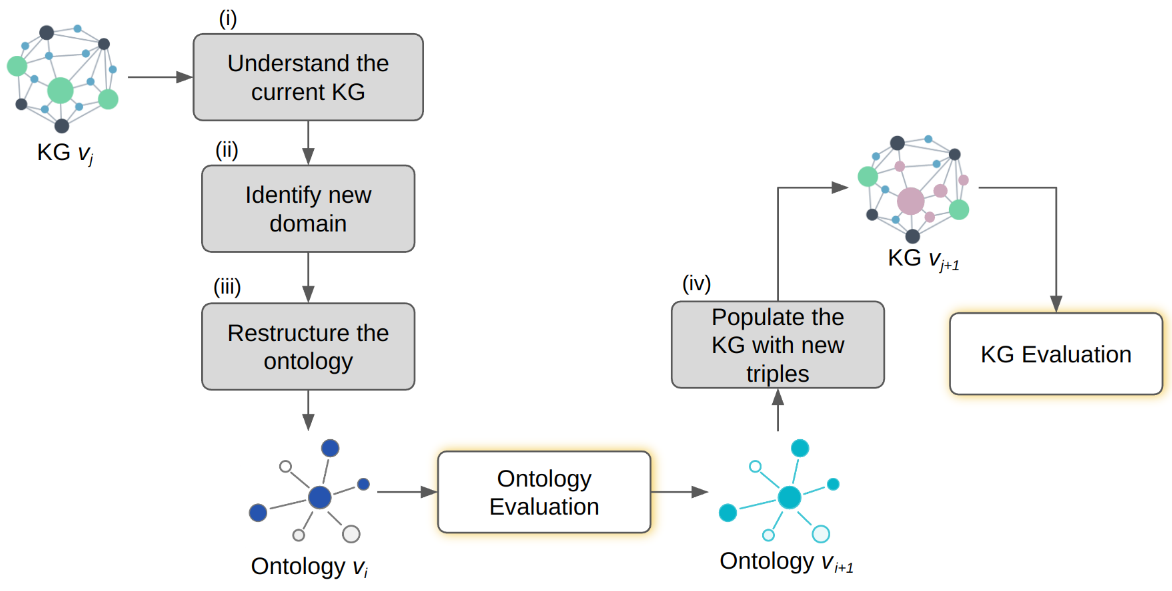


Figure 5.1: Approaches to the evaluation of ontology and the generated KG. The ontology evaluation was performed at the end of the third module of UpKG, whereas the KG evaluation was performed at the end of the framework execution.

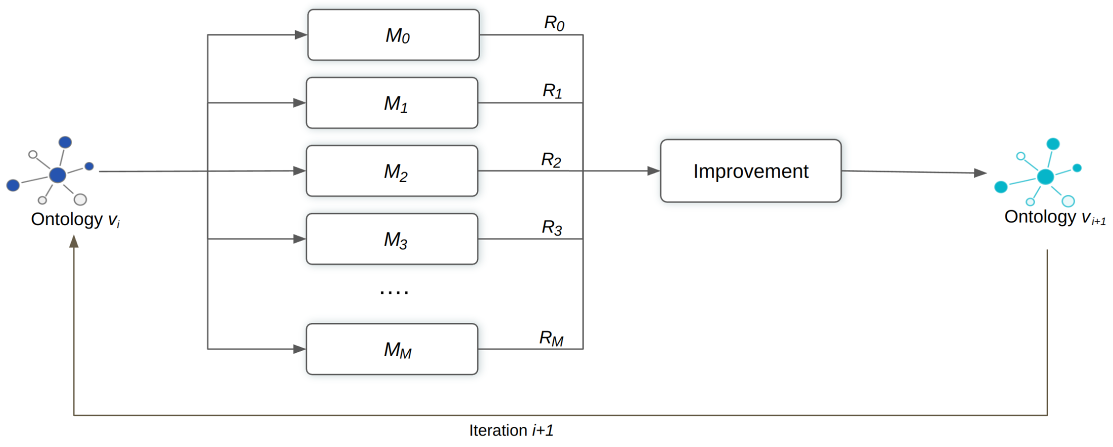


Figure 5.2: UpOntology is a new approach to ontology evaluation, the diagram represents an evaluation of ontology by iterations, where each iteration applies M previously selected methods. In addition, an improvement process is performed in each iteration to alleviate the errors found for each method M_j.

5.1.1 Ontology Evaluation

The evaluation of an ontology is a fundamental process within the ontology engineering and knowledge management fields. Its importance lies in several key aspects that directly affect the effectiveness and usefulness of created ontology in various applications. Evaluation ensures the quality and accuracy of the ontological model. A well-defined ontology must accurately and consistently represent the knowledge of the domain it encompasses. Evaluating ontology can identify potential errors, which is essential to maintain the integrity of the information stored.

In addition, the usability of ontology is a critical factor and helps determine its ability to be used in practical applications. This includes ease of search and retrieval of information. A valid and coherent ontology is essential for applications based on it to be effective and useful in real-world contexts. In other words, evaluating an ontology is critical to ensure its quality, coherence, consistency, and usefulness in real-world applications.

First, the coherence and consistency of ontology are key elements to evaluate. This means ensuring no contradictions or ambiguities in the definitions of concepts and relationships. Coherence and consistency are essential to maintain the integrity of the represented knowledge and avoid conflicting interpretations. Properly classifying concepts in the ontological hierarchy is also crucial to evaluate. Each concept must be in the right category and connected consistently with other ontology-related concepts.

As we observed earlier, ontology evaluation is a fundamental process to ensure the quality of our ontology. We propose *UpOntology*, an new evaluation approach based on iterations as shown in Figure 5.2.

This approach takes as input an initial ontology `vi`, which is evaluated by M evaluation methods in parallel, where each evaluation method (M_j) gives us results according to the characteristics we are evaluating independent of the other methods. Once we have the results R_j from each method M_j used, the next step in our *UpOntology* approach is to perform an improvement step. The improvement step shown in Figure 5.2 consists of making all the necessary improvements to the ontology `vi` depending on the results obtained by each evaluation method. As a result of this improvement process, we have a new ontology version updated `vi+1` that is returned as input for a new iteration.

We emphasize that this approach takes N iterations, where N represents the number of iterations necessary to ensure that our ontology is coherent and consistent, while ensuring the correct knowledge representation within the ontology.

We used three evaluation methods studied in the literature to evaluate our ontology, resulting from the third module using *UpOntology*. The methods used are the *Reasoner Pellet*, *OOPS*!, and *OntoDebug* as shown in Figure 5.3.

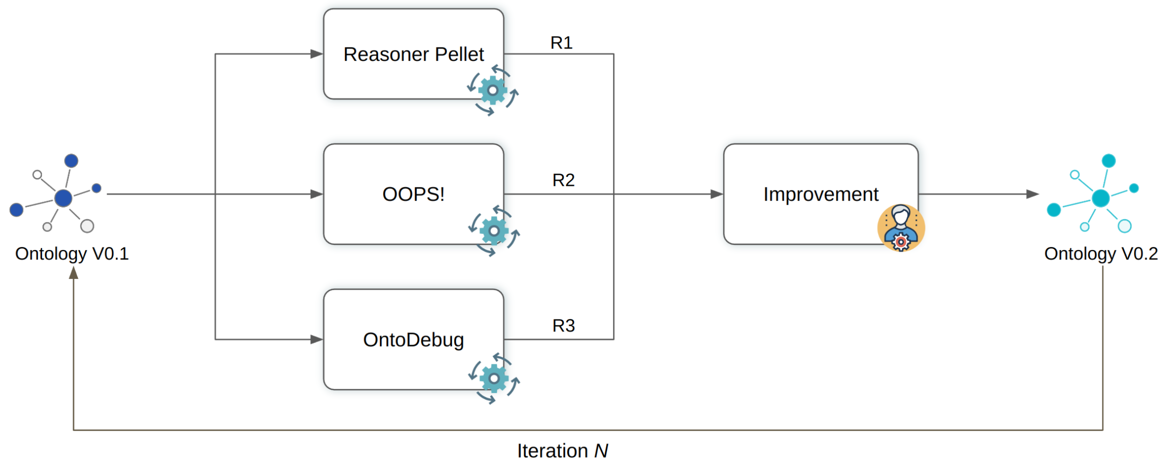


Figure 5.3: UpOntology instantiated with *Pellet, OOPS! and Ontodebug* evaluation methods. Flow to evaluate ontology 5.5 using the *UpOntology* approach.

5.1.2 Knowledge Graph Evaluation

Evaluating a KG is important because it guarantees the quality of the structured data and the precision of the relations between entities. The KG must be able to represent the information in a coherent and precise manner, which in turn improves understanding and interoperability between systems and applications.

The lack of evaluation can lead to incorrect data or ambiguous relationships in the KG, which could harm the decision-making and reliability of applications that depend on this data. The KG evaluation allows us to measure its relevance, coverage, and response level for a specific domain. This allows us to determine if it meets its specifications and provides accurate answers.

The purpose of evaluating the KG resulting from the application of the UpKG framework allows us to verify the quality of the answers generated by the new version of the KG, ensuring that it can provide accurate and relevant answers to users' questions. In addition, this approach seeks to validate that the "Appliance" domain was inserted (encoded) accurately in the KG, maintaining its integrity in relation to the existing domain of "Automobiles".

Competency Questions (CQ) [35] are questions in natural language that describe the level of knowledge represented by an ontology. An ontology should represent these questions using its own terminology and describe the answers using axioms and definitions. Competence Questions define the requirements for an ontology and are mechanisms that characterize the search space for the design of an ontology. In addition, CQs are used to evaluate ontological relationships and identify if ontology meets the desired requirements. Questions are asked to assess capacity, ability or knowledge in relation to a specific domain set. We apply CQ in our KG evaluation approach to verify that the responses returned by KG are similar to the expected responses.

5.2 Evaluation Methods

In this section, we proceed to detail the methods used in evaluating our ontology and KG.

5.2.1 Ontology

- **Reasoner Pellet [32]:** This tool is a logic-based reasoning engine. Pellet aims to process and infer semantic data represented in the OWL ontology language. In addition, this tool implements advanced reasoning algorithms that extract logical conclusions from semantic data and ontologies, which is important for constructing knowledge systems and applications in the semantic web. This tool enriches the semantic meaning of data, facilitates the resolution of more complex queries, and can make deductive inferences.

- **OntOlogy Pitfall Scanner! (OOPS!) [27]:** This tool was designed to identify and detect common errors in ontologies automatically. Its main function is to help ontologists and ontology designers refine and improve their semantic models by highlighting existing ontologies' possible problems, inconsistencies, or ambiguities.

OOPS! uses reasoning and logical analysis techniques to evaluate ontologies and generate detailed reports on problem areas, allowing users to correct and optimize their ontologies for better performance and understanding.

- **OntoDebug [30]:** It is a Protégé plugin to debug and improve ontologies and knowledge models. This plugin provides a set of specific tools and functionalities for detecting and solving common problems in ontologies. This allows ontologists and ontology designers identify inconsistencies, modeling errors, and ambiguities in their ontologies, which is essential to ensure the quality and usability of knowledge models on the Semantic Web field.

5.2.2 Knowledge Graph

- **Competency Questions (CQs):** We proposed to apply Competency Questions (CQ) [14], which are questions designed to evaluate whether the KG can correctly respond to queries related to a specific domain. To conduct this evaluation, we first defined a collection of CQs covering key aspects of the domain the KG aims to represent. Afterwards, we constructed and applied SPARQL queries (one mapped to each CQ) to retrieve from the KG responses to the collection of CQs raised beforehand. The responses obtained are compared with expected responses for each CQ. This allows us to evaluate the quality and precision of the KG responses.

  To perform the evaluation of the KG through CQs it is necessary to follow a defined flow in Figure 5.4, from obtaining the questions to generating SPARQL queries related to the question. Below we show the necessary steps to carry out the KG evaluation:

  1. Manually collect a sample of questions and answers representing the knowledge stored in the KG.
  2. Automatic extraction of entities based on the question domain.
  3. Generate SPARQL queries manually using the extracted entities.
  4. Automatic execution of the query in the triplestore (GraphDB).
  5. Manually compare the answer obtained from triplestore with the real answer given by a virtual assistant.
  6. Validation taking into account the number of questions that KG can answer.

5.3 Experimental Setup

We detail the design and configuration of our experimental setup to assess the effectiveness of our UpKG framework.

To evaluate our ontology several evaluation methods were taken into account (cf. Figure 5.3). It is important to clarify the state of ontology `v0.1`. Ontology `v0.1` is the result of applying the third module of our UpKG framework. This ontology version

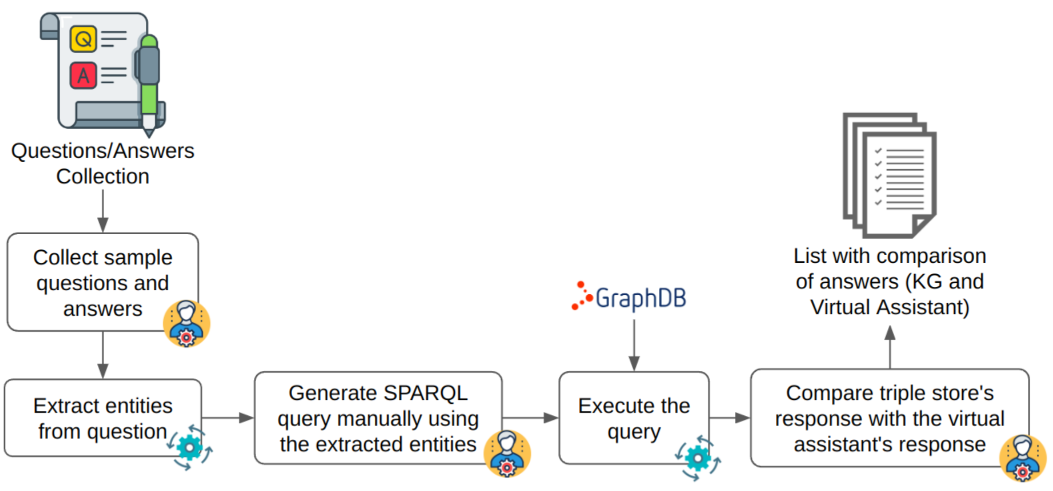

Questions/Answers Collection
Collect sample questions and answers
Extract entities from question
Generate SPARQL query manually using the extracted entities
GraphDB
Execute the query
Compare triple store's response with the virtual assistant's response
List with comparison of answers (KG and Virtual Assistant)

Figure 5.4: Flow to evaluate KG through CQs. In the first step we collect a sample of questions and answers with which we will evaluate the KG, the second step is responsible for extracting the entities of the question, in the third step we generate the SPARQL queries, in the fourth step we execute the queries generated in the *triplestore* and finally we compare the responses obtained with the real answers.

already has defined the properties and relations which are essential to encompass the new domain to be newly inserted in the KG. In addition, this ontology is fully integrated with the existing ontology of GoBots, which guarantees a connection between the different domains already defined and contributes to a unified and enriched knowledge base.

While the third module of the UpKG framework generates a new version of an ontology, as exemplified in Figure 5.1, we cannot ensure that this is completely correct despite following the best practices when developing such ontology. To this end, we applied the UpOntology approach with two iterations to evaluate our initial ontology, which we denote as `v0.1`. It is important to highlight that the application of the UpOntology approach makes use of N iterations necessary to guarantee the quality of the ontology, knowing this number in advance is impossible, but it is only known when all the evaluation methods do not present errors in the $i-th$ iteration. For example, all evaluation methods presented errors in the first iteration. However, once the errors are corrected in the improvement stage, we proceed to evaluate the ontology with a second iteration, and in this execution we no longer present errors. That is why in our evaluation approach only two iterations were necessary to guarantee the coherence and consistency of our ontology.

It is important to note that the *Pellet* and *OntoDebug* evaluation methods were carried out within the Protégé software and the OOPS! [1] within its Web system.

The evaluation of the KG was taken into account through CQs. It is important to clarify the status of KG. The final KG is the result of applying the fourth module of our UpKG framework. This KG is already populated with information (triples) from various pairs of questions and answers that have been transformed into triples and inserted at the stage of this fourth module.

In particular, this KG version contains information from different domains as previ-

[1] `https://oops.linkeddata.es/index.jsp`

ously explained in section 4.2.4 (considering the context of our case study), specifically were processed pairs of questions and answers of five merchants “Appliances" domain and five merchants “Automobiles" domain, having a total of 3382 new instances.

This KG was hosted on an instance of the GraphDB [2], a semantic graph database (RDF Store). GraphDB allows us to perform SPARQL queries easily and is where we run the assessment of CQs.

5.4 Results

We present the results of the ontology evaluation and the KG evaluation.

5.4.1 Results of the Ontology Evaluation

We present the results of applying Pellet, OOPS! and Ontodebug to evaluate coherence and consistency using the UpOntology approach. We considered N iterations when applying this approach (cf.Figure 5.3); the output of the first iteration was the input for the second iteration.

In the following, we present the results of the two executions for Pellet, OOPS! and OntoDebug.

First execution

We show the results of applying the first iteration of the evaluation module by iterations. We had as input the ontology `v0.1` shown in Figure 5.5. Ontology `v0.1` was the result of applying our proposed UpKG framework to the third module (cf. Figure 3.1). For this reason, ontology `v0.1` is used as input for each evaluation method below.

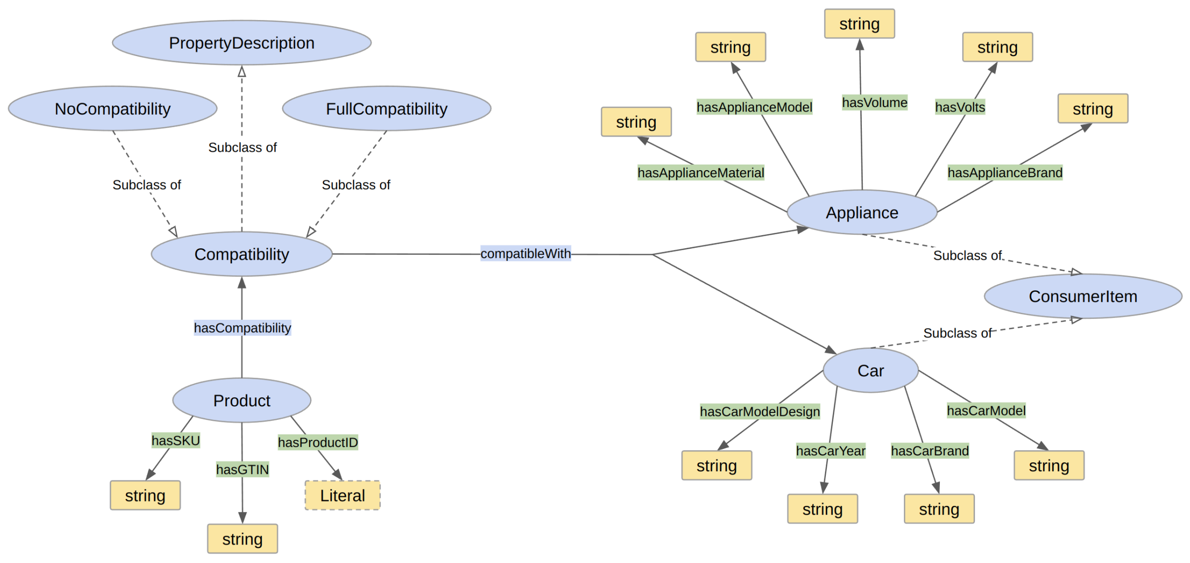
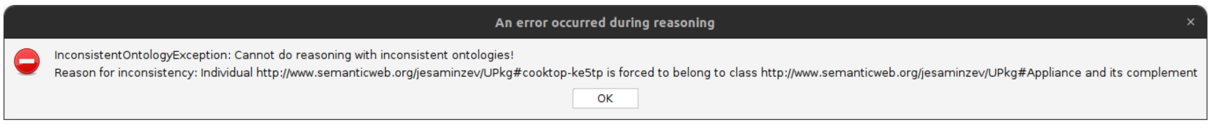


Figure 5.5: Ontology `v0.1`

[2]https://graphdb.ontotext.com/

1. **Reasoner Pellet**

   Figure 5.6 presents the results of applying Pellet to evaluate the consistency of the input ontology, which is indicating that the evaluated ontology still has errors and it is not consistent under this reasoner. Additionally, Figure 5.6 presents a message indicating that the error is due to the fact that an instance of type Car cannot be of type Appliance at the same time.

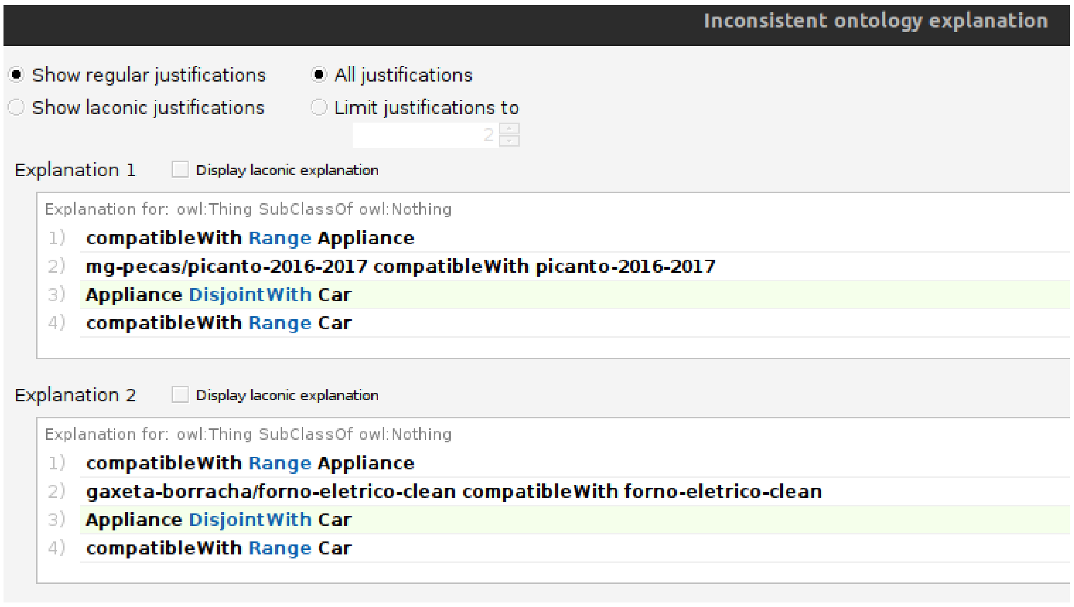


An error occurred during reasoning

```
InconsistentOntologyException: Cannot do reasoning with inconsistent ontologies!
Reason for inconsistency: Individual http://www.semanticweb.org/jesaminzev/UPkg#cooktop-ke5tp is forced to belong to class http://www.semanticweb.org/jesaminzev/UPkg#Appliance and its complement
```

Figure 5.6: Reasoner Pellet inconsistent

For a better understanding of the inconsistency error, it is possible to access the logs that Protege gives us when we start the reasoner. Figure 5.7 shows such logs, a list of all instances that make ontology inconsistent. Figure 5.7 presents several cases of the error found, where an instance of the Compatibility type is related to the “Appliance" class. However, the reassignment infers that this relationship can also be with the ‘Car’ class of ontology `v0.1`. That is where our ontology is inconsistent because the ‘Appliance’ and ‘Car’ classes are disjointed. For example, the instance `gaxeta-drunk/forno-eletrico-clean` is compatible with the instance Appliance `forno-eletrico-clean`. Still, the reasoner infers that `forno-eletrico-clean` may be of the type ‘Car’.



Inconsistent ontology explanation

● Show regular justifications ● All justifications
○ Show laconic justifications ○ Limit justifications to 2

Explanation 1 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing
1) compatibleWith Range Appliance
2) mg-pecas/picanto-2016-2017 compatibleWith picanto-2016-2017
3) Appliance DisjointWith Car
4) compatibleWith Range Car

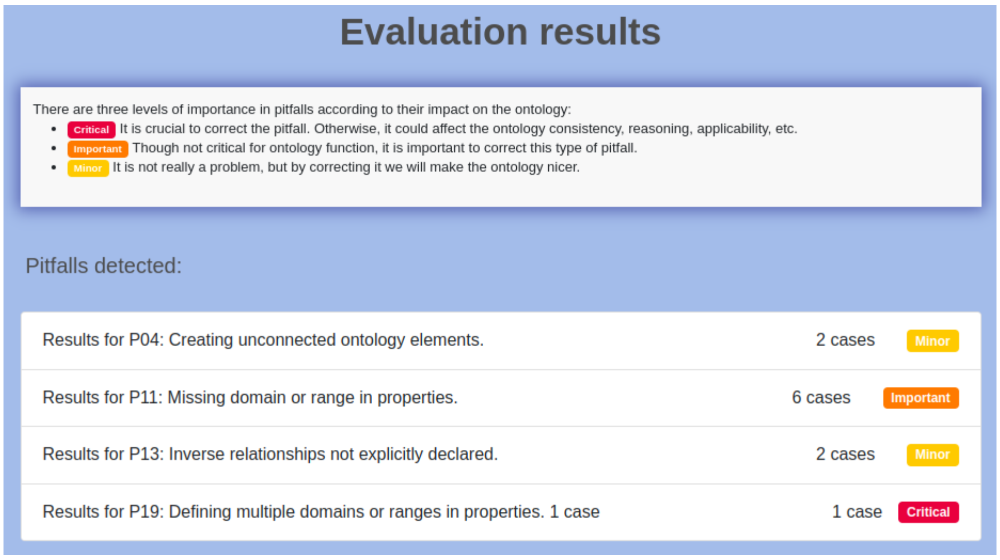
Explanation 2 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing
1) compatibleWith Range Appliance
2) gaxeta-borracha/forno-eletrico-clean compatibleWith forno-eletrico-clean
3) Appliance DisjointWith Car
4) compatibleWith Range Car

Figure 5.7: Reasoner Pellet inconsistent summary

2. **OOPS!**

   To evaluate our ontology `v0.1`, we used the ontology error scanner (OOPS!). Figure 5.8 shows the result of the evaluation executed by the OOPS! tool. There is a critical alert in multiple domains or ranges in the list of detected pitfalls. This alert concerns the `compatibleWith` property causing ontology inconsistency. In addition, there is an important alert due to the lack of domain or range in properties, and finally two minor alerts are the creation of disconnected ontology elements and undeclared inverse relationships.


Evaluation results

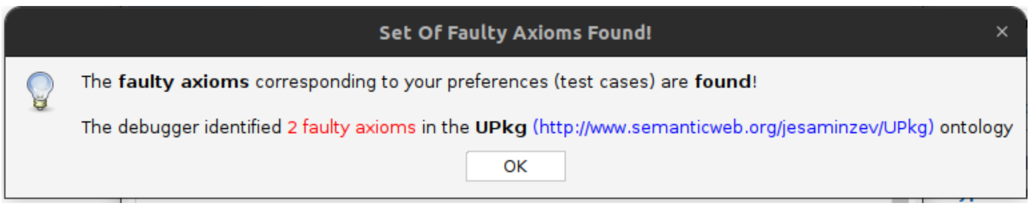
There are three levels of importance in pitfalls according to their impact on the ontology:
- Critical It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- Important Though not critical for ontology function, it is important to correct this type of pitfall.
- Minor It is not really a problem, but by correcting it we will make the ontology nicer.

Pitfalls detected:

Results for P04: Creating unconnected ontology elements.	2 cases	Minor
Results for P11: Missing domain or range in properties.	6 cases	Important
Results for P13: Inverse relationships not explicitly declared.	2 cases	Minor
Results for P19: Defining multiple domains or ranges in properties. 1 case	1 case	Critical

Figure 5.8: List of pitfalls detected in the ontology `v0.1` applying *OOPS*!

3. **OntoDebug**
The structure and relationships of the ontology `v0.1` were validated using Protégé built-in debugging tool OntoDebug[3]. The main goal of OntoDebug is to help find those axioms in ontologies, giving information if an ontology is inconsistent. Figure 5.9 presents the OntoDebug output of ontology `v0.1`. The results show that there are two faulty axioms for this reason our ontology is inconsistent.


Set Of Faulty Axioms Found!

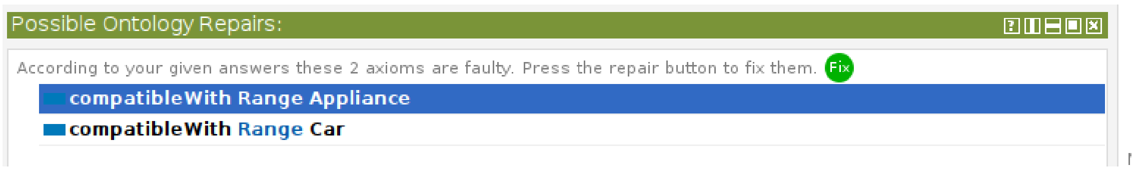
The **faulty axioms** corresponding to your preferences (test cases) are **found**!

The debugger identified 2 faulty axioms in the **UPkg** (http://www.semanticweb.org/jesaminzev/UPkg) ontology

Figure 5.9: The debug output of the UpKG ontology `v0.1` from *OntoDebug* tool showing that there are two faulty axioms.

Figure 5.10 shows the two faulty axioms present in the ontology `v0.1`. The inconsistency of the ontology is due to the fact that an instance is inferred as *`compatibleWith`* `Range 'Appliance'` and *`compatibleWith`* `Range 'Car'`, but at the same time `Appliance` *`disjointWith`* `Car` and `Car` *`disjointWith`* `Appliance`. In other words, the instance has 'Appliance' and 'Car' classes as range. Nevertheless, the instance cannot be both classes simultaneously because these classes are disjoint.

[3]http://isbi.aau.at/ontodebug/


Possible Ontology Repairs:

According to your given answers these 2 axioms are faulty. Press the repair button to fix them.

compatibleWith Range Appliance

compatibleWith Range Car

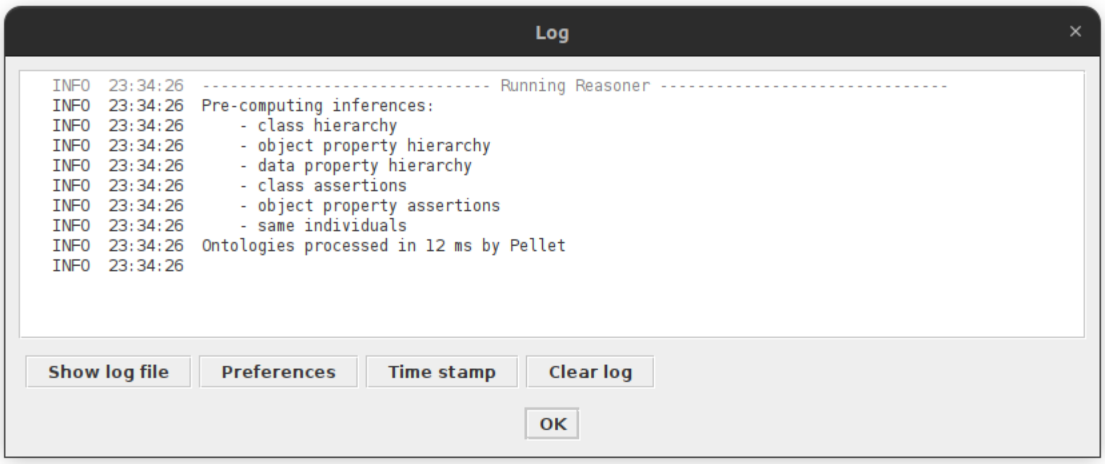
Figure 5.10: The two faulty axioms in the ontology `v0.1` – *OntoDebug* tool

Second execution

We present the results of applying the second iteration of the ontology evaluation. For this iteration, the input data was the ontology `v0.2`, which contains all the corrections performed from the first iteration.

1. **Reasoner Pellet**

   In this iteration, the Pellet reasoner shows that the ontology `v0.2` is consistent since it does not present errors when evaluating it (cf.Figure 5.11).

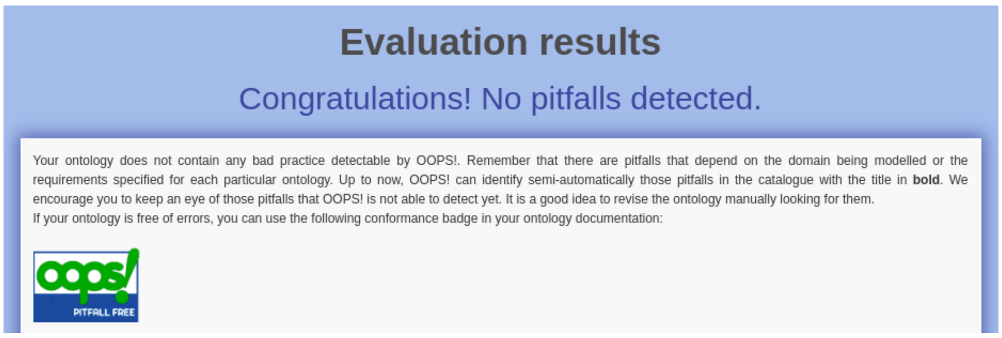


```
INFO  23:34:26  ------------------------------ Running Reasoner ------------------------------
INFO  23:34:26  Pre-computing inferences:
INFO  23:34:26      - class hierarchy
INFO  23:34:26      - object property hierarchy
INFO  23:34:26      - data property hierarchy
INFO  23:34:26      - class assertions
INFO  23:34:26      - object property assertions
INFO  23:34:26      - same individuals
INFO  23:34:26  Ontologies processed in 12 ms by Pellet
INFO  23:34:26
```

Figure 5.11: Logs of applying the *Pellet* in the second evaluation iteration

2. **OOPS!**

   Figure 5.12 shows the evaluation results of the *OOPS*! tool. The result shows that ontology `v0.2` does not present pitfalls in this second execution.



Evaluation results

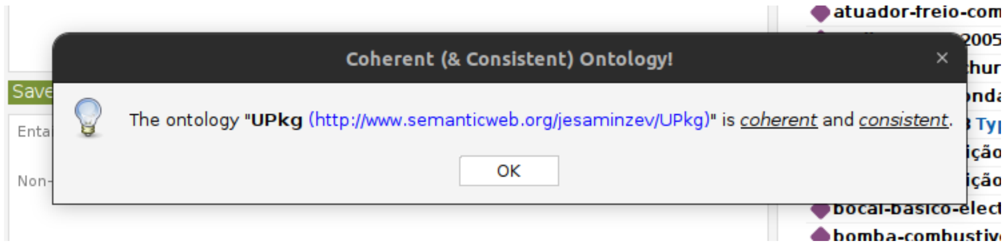
Congratulations! No pitfalls detected.

Your ontology does not contain any bad practice detectable by OOPS!. Remember that there are pitfalls that depend on the domain being modelled or the requirements specified for each particular ontology. Up to now, OOPS! can identify semi-automatically those pitfalls in the catalogue with the title in **bold**. We encourage you to keep an eye of those pitfalls that OOPS! is not able to detect yet. It is a good idea to revise the ontology manually looking for them.
If your ontology is free of errors, you can use the following conformance badge in your ontology documentation:

Figure 5.12: Outcome of the second iteration in applying the *OOPS*! tool

3. **OntoDebug**

Figure 5.13 presents our ontology `v0.2` validated with *OntoDebug*. We obtained as a result of the execution, that our ontology is coherent and consistent.



Coherent (& Consistent) Ontology!

The ontology "**UPkg** (http://www.semanticweb.org/jesaminzev/UPkg)" is coherent and consistent.

Figure 5.13: The debug output of the UpKG ontology shows that it is consistent and coherent - in the second iteration

At the end of the second iteration, we can affirm that our ontology was consistent and coherent (Figure 5.11 and Figure 5.13). In addition, our ontology did not present pitfalls (cf. Figure 5.12).

Table 5.1 represents a behavior summary during executions used in ontology evaluation. This table provides key information on ontology performance and quality.

Table 5.1: Summary of the behavior of the executions for the ontology evaluation. The second column shows the results obtained in the first execution applying the Pellet, OOPS! and OntoDebug methods. The third column shows the results obtained in the second execution after improving ontology with the observations of the first execution.

	First Execution	**Second Execution**
Pellet	✗	✓
OOPS!	✗	✓
OntoDebug	✗	✓

5.4.2 Results of the Knowledge Graph Evaluation

This section shows the results of applying our evaluation to the resulting KG, which contains 1324 instances of the *Compatibility* type; 995 instances of the *ConsumerItem* type; and finally, 1063 of the *Product* type.

Competency Questions

To evaluate our KG `v1` using the CQs, it was necessary to have a set of questions that SPARQL queries might answer. These CQs were extracted from the same questions that populated our KG in the fourth module of our UpKG framework to verify if the KG responds adequately to these previously inserted questions. The CQs were compiled manually, in which we extracted three random questions from each merchant. In addition, the generation of SPARQL queries was performed manually taking into account the extraction of entities from questions that were randomly chosen. These SPARQL queries were then executed in the environment where the KG `v1`is stored, then we compared the actual response of the human assistants with the response obtained from the KG `v1`. Table 5.2 presents the thirty questions that were satisfactorily answered.

Our updated KG `v1` stores information from two domains ('Car' and 'Appliances'), which were extracted from ten different merchants presented in Gobots client base; five merchants contain questions and answers to the 'Appliances' domain; and the other five merchants contain questions and answers of 'Automobiles' domain.

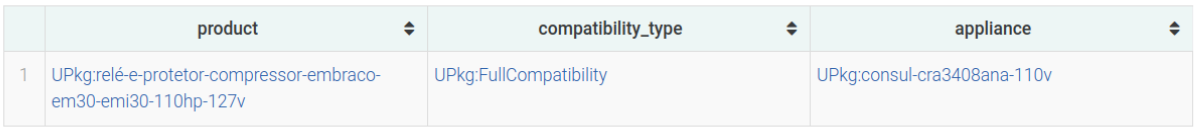
In the following, we present four questions out of thirty CQs, of which two were intended to demonstrate that information can be extracted from the 'Appliances' domain and another two from the 'Automobiles' domain. This information extraction was done through SPARQL queries.

Each item presented describes five parts within each CQ:

- **Question** is the question asked by a customer in the e-commerce system;
- **Answer** is the answer given by a smart human assistant from each store;
- **Product** is the product information from which they are asking the question;
- **SPARQL** is the query generated from the question asked;
- **GraphDB** is the KG's response to the generated SPARQL.

1. Competency Question #01:
   - Question: Serve na geladeira CRA3408ANA CONSUL 340 LITROS 110V?
   - Answer: Sim, serve no compressor Embraco de sua Cônsul, sim.
   - Product: MLB1632141487
   - Sparql:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
        UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1632141487"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Consul";
               UPkg:hasApplianceModel "CRA3408ANA";
               UPkg:hasVolts "110V".
}
```



	product	compatibility_type	appliance
1	UPkg:relé-e-protetor-compressor-embraco-em30-emi30-110hp-127v	UPkg:FullCompatibility	UPkg:consul-cra3408ana-110v

Figure 5.14: Response to CQ1 from KG within the GraphDB

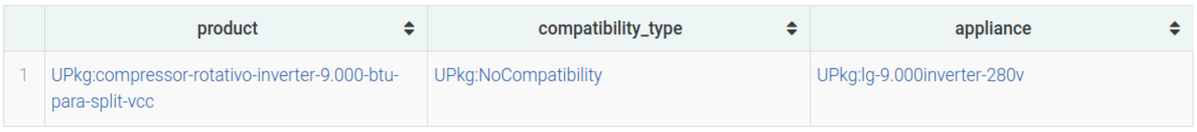
- GraphDB:

2. Competency Question #02:

- Question: Boa tarde serve no LG 9.000 inverter 280 v.
- Answer: B tarde. Para LG é outro o modelo, de tensão maior.
- Product: 7897738500192
- Sparql:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7897738500192"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "LG";
               UPkg:hasApplianceModel "9.000 inverter";
               UPkg:hasVolts "280v".
}
```

- GraphDB:



	product	compatibility_type	appliance
1	UPkg:compressor-rotativo-inverter-9.000-btu-para-split-vcc	UPkg:NoCompatibility	UPkg:lg-9.000inverter-280v

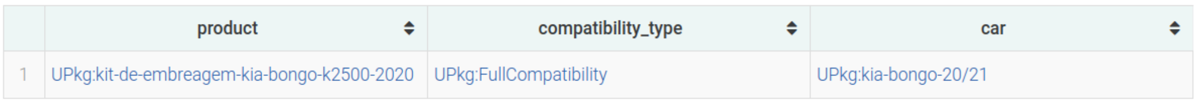
Figure 5.15: Response to CQ2 from KG within the GraphDB

3. Competency Question #03:

- Question: Serve na KIA BONGO 20/21?
- Answer: Boa tarde, serve sim.
- Product: MLB2719751431
- Sparql:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2719751431"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "kia";
         UPkg:hasCarModel "Bongo";
         UPkg:hasCarYear "20/21".
}
```

- GraphDB:



	product	compatibility_type	car
1	UPkg:kit-de-embreagem-kia-bongo-k2500-2020	UPkg:FullCompatibility	UPkg:kia-bongo-20/21

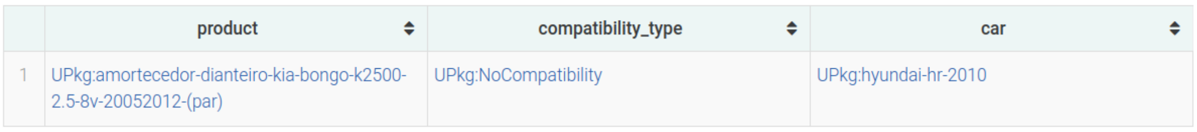
Figure 5.16: Response to CQ3 from KG within the GraphDB

4. Competency Question #04:

- Question: Serve na HYUNDAI HR 2010????
- Answer: Boa tarde! esse não serve.
- Product: MLB2761007508
- Sparql:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB3387033577"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "hyundai";
         UPkg:hasCarModel "HR";
         UPkg:hasCarYear "2010".
}
```

- GraphDB:



	product	compatibility_type	car
1	UPkg:amortecedor-dianteiro-kia-bongo-k2500-2.5-8v-20052012-(par)	UPkg:NoCompatibility	UPkg:hyundai-hr-2010

Figure 5.17: Response to CQ4 from KG within the GraphDB

Table 5.2 summarizes the CQs executed in KG `v1`. This Table includes the questions and answer pairs used to populate our KG `v1` and the answers obtained from KG `v1`. In addition, Table 5.2 has the Status column that verifies if the KG `v1` response equals the response collected from the GoBots database.

Appendix A presents the complete results for the other CQs.

5.5 Discussion

This Chapter described an important aspect of our research work: the evaluation of ontology and KG resulting from applying the UpKG framework. This evaluation process was essential to ensure the quality and usefulness of the knowledge represented and stored in the KG, especially in e-commerce, where accuracy and responsiveness are critical.

This study proposed two evaluation approaches: (i) Ontology Evaluation and (ii) KG Evaluation. These approaches were important because they might ensure that the structure of ontology as the data encoded in the KG are consistent and of high quality.

Ontology Evaluation focused on verifying the consistency and coherence of the ontology structure, whereas the KG Evaluation emphasized evaluating the quality of the data that make up the KG and the quality of the responses provided by it via SPARQL queries.

To evaluate the ontology, we implemented *UpOntology*, an iterative approach incorporating three evaluation methods: *Pellet, OOPS! and OntoDebug*. In the first execution of the evaluation of our ontology, all three methods showed inconsistencies. This result served to improve the model and corrections were made. In the second execution (iteration), the three methods did not identify any inconsistencies. This result indicated that our ontology achieved a satisfactory level of consistency and consistency, which validates the effectiveness of our UpOntology approach.

Regarding the KG evaluation, we applied a set of CQs in the 'Appliances' and 'Automobiles' domains, representing critical and relevant questions. These CQs were derived from the original questions that populated the KG in the 'Appliances' and 'Automobiles' domains and were used to assess the KG's ability to respond accurately and adequately.

The evaluation process allowed us to identify which parts require improvements and adjustments and, in turn, contribute to a precise and effective updated KG.

5.6 Final Remarks

This Chapter described the evaluation of our UpKG framework. We developed two evaluation approaches, one focused on ontology and the other on KG, allowing us to evaluate the consistency of ontology and the quality of KG responses. Through these evaluations,

Table 5.2: Summary table of CQ executed in KG v1. The second and third column are the question and answer pairs collected from the Appliances and Automobiles domain, the fourth column shows the answer obtained from the KG v1 which is stored in GraphDB, the fifth column shows if the answer obtained from the KG v1 is the same to the answer of the assistant.

ID	Question	Answer	Answer KG v1	Status
1	Boa tarde. Essa escova serve para o modelo ERG22?	Sim, esta escova serve e é compatível com erg22	Compatible	✓
2	Serve na electrolux ll08s? caso negativo, vc teria modelo que sirva na ll08s?	Sim serve para o modelo LL08S.	Compatible	✓
3	Serve no Eletrolux Eco turbo?	Sim, compatível.	Compatible	✓
4	Serve Mr Coffee VB35 ?	Resistência 127V compatível com Cafeteira Programável.	Compatible	✓
5	Boa tarde! Esta Jarra é compatível ao Liquidificador Oster Modelo BVLB07-L00-057	Boa tarde. anunciado é compatível com o modelo informado	Compatible	✓
6	Boa tarde Serve panela oster modelo 8030	Não é compatível com o modelo 8030	Incompatible	✓
7	Boa tarde, esse copo serve para o modelo L-1000 BI 12V ?	Sim, compatível.	Compatible	✓
8	O copo de liquidificador serve no Mondisl L 1000W ?	Sim, compatível.	Compatible	✓
9	Ele serve no modelo special line l48	Não é compatível.	Incompatible	✓
10	Serve na geladeira CRA3408ANA CONSUL 340 LITROS 110V?	Sim, serve no compressor Embraco de sua Cônsul, sim.	Compatible	✓
11	Boa tarde serve no LG 9.000 inverter 280 v	Para LG é outro o modelo, de tensão maior.	Incompatible	✓
12	Bom dia! Serve no compressor Embraco EM2X3134U (127V)?	A linha EM utiliza outro modelo.	Incompatible	✓
13	Serve para Gaggia Anima Panarello?	Sim, serve.	Compatible	✓
14	Boa tarde, esse interruptor serve no ASPIRADOR CYCLONE ELECTROLUX TIT10 1450 W ?	Boa tarde! Sim, serve.	Compatible	✓
15	Da certo no processador Philips série 3000 Potencoa 600w Ri7300?	Sim, serve	Compatible	✓
16	Serve na KIA BONGO 20/21?	Boa tarde , serve sim.	Compatible	✓
17	Ela serve para bomho k2700 4x4 ano 2009?	Serve sim.	Compatible	✓
18	Serve na HYUNDAI HR 2010?	Boa tarde! esse não serve.	Incompatible	✓
19	Desculpa... não sei se corresponde. Seria para o pneu fino de estepe do HB20 2019.	Olá PIUI171. É adequado para o uso em um HB20 2019.	Compatible	✓
20	Boa tarde Esse pneu pode ser utilizado na Santa Fé 2008 Os que eu tenho na Santa Fé são 235 60 18	Se as medidas do produto correspondem às utilizadas atualmente deverá atender sim.	Compatible	✓
21	Boa tarde. Serve na Frontier Attack 2019?	Se as medidas do produto correspondem às utilizadas atualmente deverá atender sim.	Compatible	✓
22	Serve no Siena fire 1.4 8v 2015? Obs não é motor evo	Olá! Tudo bem? Não serve.	Incompatible	✓
23	serve na palio weekend adventure 1.6 16v ano 2002/03? grato	Olá! Tudo bem? Serve sim.	Compatible	✓
24	Serve na ranger 2008 2.3 gasolina ?	Sim, serve na Ranger 2008 2.3 gasolina.	Compatible	✓
25	9bwab45u4kt047833 serve para este gol 2018/19. 1.6 msi	Bom dia, infelizmente nao da certo e nao temos a peça correta.	Incompatible	✓
26	Esse comando serve no gol g5 2010 1.0 chassi 9BWAAO5UOBT136830	Aplica perfeitamente em seu veículo.	Compatible	✓
27	Serve na Titan 2007 es?	Serve sim	Compatible	✓
28	Essa peça da certo na Titan 150 /2007	Serve perfeitamente!	Compatible	✓
29	Serve na g310 2021?	Serve perfeitamente!	Compatible	✓
30	Esse quite serve pra Twister 2021	Negativo. Serve para Todas CB 300 Apenas.	Incompatible	✓

we demonstrated the consistency and coherence of our updated ontology and that our outcome KG could provide accurate answers in the two defined domains. The next Chapter presents the contributions and conclusions of this Dissertation and explores directions for future research.

Chapter 6

Conclusion

This Chapter presents the conclusions of this M.Sc. dissertation around our UpKG framework and its implementation (application) in the context of e-commerce. In addition, the contributions arising from this study and future studies are highlighted.

Section 6.1 focuses on describing the achievements reached in our study. We provide a review of our achievements at each stage of our study, highlighting the theoretical and practical aspects that contribute to our research work. In addition, research questions defined in chapter 1 are answered. Section 6.2 presents and analyzes the contributions obtained as a result of the exhaustive development of this research. Section 6.3 discusses future studies that arise from the findings and results obtained in this research work. These future studies describe new opportunities to improve and expand the UpKG framework.

6.1 Synthesis of Achievements

This research work obtained significant achievements in the conceptualization, design, application, and evaluation of the UpKG framework to insert new domains in a KG.

Our study focused on a comprehensive review of the literature related to KGs in the context of e-commerce. Existing methodologies and tools were identified and analyzed, highlighting the limitations and challenges they face. This review of related studies provided the necessary conceptual basis to justify the need for a new approach, such as the UpKG framework.

We presented a detailed description of the UpKG framework. This description of the framework was cornerstone as it allowed us to understand the architecture of the UpKG, highlighting the essential modules from understanding the current KG to populating the updated version of a KG. The conceptualization and integration of these modules were addressed in depth, providing a clear view of how our UpKG framework allows inserting a new domain.

We developed a practical application of the UpKG framework in a real scenario, specifically in the GoBots company. We managed to detect the new domains to insert the KG, restructure the existing ontology, and successfully insert triples of ‘Appliances’ domain in the KG. In addition, the results obtained during the application show the effectiveness of the UpKG in practice.

Finally, we conducted an ontology and KG evaluation resulting from the UpKG framework. We applied two approaches focused on ontology and KG to evaluate the consistency and coherence of ontology, as well as the quality of KG responses. The evaluation results confirm the coherence and consistency of our ontology. These findings were derived from the application of three different evaluation methods: *Pellet, OOPS!, and OntoDebug.* In addition, the KG evaluation results demonstrated that the KG `v1` was capable of providing accurate answers in the Appliance and Automobile domains.

We presented two research questions in Chapter 1, and we provide the following answer relying on the research conducted in this Dissertation:

1. *How can the UpKG framework can be suited to the constant information evolution in e-commerce, allowing the insertion of new domains without compromising the coherence and integrity of the underlying KG in place?*

   UpKG framework adapts to the evolution of information in e-commerce and the insertion of new domains, without compromising the coherence and integrity of the existing KG. Our framework has a modular design which allows easy adaptation to changes. The flexibility of the KG ontology is achieved by identifying the entities and relationships associated with new domains. This approach ensures that the introduction of these elements does not compromise the coherence of the KG. In addition, UpKG framework has a sequential approach to evaluating and improving the KG. Updates are made in a controlled manner, which allows for correction and adjustment in case coherence problems arise during the insertion of new domains.

2. *How does inserting new KG domains through our proposed UpKG framework impact end-user queries on e-commerce platforms?*

   Inserting new domains into a KG through the UpKG framework has a major impact on end-user queries on e-commerce platforms. This impact is an improvement in KG's ability to provide more accurate and relevant answers to queries related to new domains. Adapting to the evolution of information, the UpKG framework facilitates the user experience and makes it more enriching by ensuring that the KG is updated and aligned with the constant needs of consumers. The insertion of new domains allows to cover of diverse and complex queries, improving the efficiency of the e-commerce platforms that use this framework.

6.2 Contributions

Our M.Sc. Dissertation achieved significant results. Our main contribution refers to the UpKG framework that guide us to insert new domains in an existing KG with a focus on e-commerce based on users' questions and answers as input. The results we obtained show that our framework allows inserting a domain to an existing KG, without compromising the coherence and consistency of the same. In summary, we present the key contributions of this research work in the following:

- The conceptual development of the UpKG framework is a key contribution as it addresses the problem of inserting new domains in KGs in the specific context of e-commerce. The proposal offers a novel solution based on the efficient management of ontologies and KGs.

- UpKG framework has an adaptability to the constant changes of information in e-commerce and this represents a significant contribution since it allows to update of the KG to accept/accommodate new domains.

- The application of the UpKG framework in the GoBots company allows us to observe the behavior of our framework in each proposed module in a real-world setting. In addition, it allowed us to ensure that our framework is flexible to accept other domains under the context of e-commerce.

- We propose an ontology evaluation methodology called *UpOntology*, which allows us to validate the applicability of the framework, as well as the flexibility and coherence of the resulting ontology. This evaluation proposal was shown as an important component of understanding the effectiveness of the framework in practical cases and real-world situations.

- This research resulted in a scientific publication as a full article at the International Conference on Knowledge Engineering and Ontology Development[1] (KEOD), where we contribute to the knowledge to the scientific community. Below is our published article:

  – **Zevallos-Quispe, J.**; Regino, A.; Chico, V.; Hochgreb de Freitas, V. and Dos Reis, J. C. (2023). *UpKG: A Framework to Insert New Domains in Knowledge Graphs.* In Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD, ISBN 978-989-758-671-2, ISSN 2184-3228, pages 85-95.

6.3 Future Studies

In this section, we propose the following future studies.

- We aim to bring the KG `v1` to a production environment to make a more rigorous and consistent evaluation. This will allow us to analyze the questions asked by users in real-time and improve KG's ability to provide accurate and contextually relevant answers. As part of this process, we plan to incorporate new domains that have been identified in the framework (module two), further enriching the knowledge in our KG.

- We will focus on the improvement of the entity's extraction step. Although the proposed tools have demonstrated solid results in this aspect, we observed that the use

[1]`https://keod.scitevents.org/Home.aspx`

of GPT family models via API in some scenarios generates questions with inaccuracies or misunderstandings, which can affect the quality of the answers. Therefore, one of our objectives in future work is to refine this tool, ensuring accurate extraction and improving the understanding of questions generated by users. This progress will allow us to provide a more effective and accurate experience in the interaction between users and KG.

- Our mission is to develop accurate and appropriate metrics that allow a thorough assessment of KG responses in varied contexts. This involves considering the accuracy, consistency, relevance, and depth of responses based on the specific characteristics of each domain. Creating robust and tailored metrics ensures that we can effectively measure the quality of responses and ultimately raise the KG performance standard.

- The application of our framework focused on user questions with the intention of Compatibility, which means a limitation when processing other types of questions. However, expanding this element to accept questions with other intentions is presented as an important direction for improvement. This expansion of intentions allows the framework to cover a greater number of questions and provide richer and more precise answers.

- Apply another evaluation technique which consists of someone else applying our UpKG framework. The main objective of adopting this evaluation technique is to determine the ease of use of our framework by external users. By providing access to individuals not previously familiar with the UpKG framework, we will be able to obtain an important evaluation of the tool's usability from a variety of perspectives. This approach will allow us to identify possible areas for improvement in the proposed stages, ensuring that the UpKG framework is accessible. In addition, the outcome of this evaluation will contribute to improving the implementation of the UpKG framework.

- We will focus on automating processes that currently require manual intervention within our UpKG framework. The objective is to minimize the dependence on manual tasks in the process, which will help optimize the efficiency of the framework. This approach seeks to not only streamline operations, but also improve accuracy and consistency by eliminating potential human errors associated with manual interventions.

- We will focus on exploring various methodologies for ontological design, among which the use of fundamental ontologies stands out. This approach involves considering solid and well-established conceptual structures, to provide a robust foundation for our representation of knowledge in UpKG.

Bibliography

[1] Ontodebug - interactive ontology debugging in protégé, 2015. `http://isbi.aau.at/ontodebug/` [Accessed: 2023-10-11].

[2] Sunitha Abburu. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, 57(17), 2012.

[3] Addi Ait-Mlouk and Lili Jiang. Kbot: a knowledge graph based chatbot for natural language understanding over linked data. *IEEE Access*, 8:149220–149230, 2020.

[4] João Barroca, Abhishek Shivkumar, Beatriz Quintino Ferreira, Evgeny Sherkhonov, and João Faria. Enriching a fashion knowledge graph from product textual descriptions. *arXiv preprint arXiv:2206.01087*, 2022.

[5] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifier (uri): Generic syntax. Technical report, 2005.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[7] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global, 2011.

[8] Tom Bocklisch, Joe Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181, 2017.

[9] ChatGPT. ChatGPT: Optimizing Language Models for Dialogue. `https://openai.com/blog/chatgpt/` [Accessed: 2023-10-25].

[10] Vinay K Chaudhri, Adam Farquhar, Richard Fikes, Peter D Karp, and James P Rice. Okbc: A programmatic foundation for knowledge base interoperability. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 600–607, 1998.

[11] Shiqian Chen, Chenliang Li, Feng Ji, Wei Zhou, and Haiqing Chen. Driven answer generation for product-related questions in e-commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 411–419, 2019.

[12] Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J Carroll, and Brian McBride. Rdf 1.1 concepts and abstract syntax. *W3C recommendation*, 25(02):1–22, 2014.

[13] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet computing*, 4(5):63–73, 2000.

[14] Fernanda Farinelli. Ontologias biomédicas: uma abordagem prática. *Fronteiras da Representação do Conhecimento*, 1(2):22–50, 2021.

[15] David C Faye, Olivier Cure, and Guillaume Blin. A survey of rdf storage approaches. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 15:11–35, 2012.

[16] Héctor Hiram Guedea-Noriega and Francisco García-Sánchez. Integroly: Automatic knowledge graph population from social big data in the political marketing domain. *Applied Sciences*, 12(16):8116, 2022.

[17] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*, 12(2):1–257, 2021.

[18] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023.

[19] Natthawut Kertkeidkachorn and Ryutaro Ichise. An automatic knowledge graph creation framework from natural language text. *IEICE TRANSACTIONS on Information and Systems*, 101(1):90–98, 2018.

[20] Aditya Khamparia and Babita Pandey. Comprehensive analysis of semantic web reasoners and tools: a survey. *Education and Information Technologies*, 22:3121–3145, 2017.

[21] Navya Martin Kollapally, Yan Chen, Julia Xu, and James Geller. An ontology for the social determinants of health domain. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2403–2410. IEEE, 2022.

[22] Feng-Lin Li, Hehong Chen, Guohai Xu, Tian Qiu, Feng Ji, Ji Zhang, and Haiqing Chen. Alimekg: Domain knowledge graph construction and application in e-commerce. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2581–2588, 2020.

[23] Shaily Malik, Anisha Goel, and Saurabh Maniktala. A comparative study of various variants of sparql in semantic web. In *2010 International Conference on Computer*

Information Systems and Industrial Management Applications (CISIM), pages 471–474. IEEE, 2010.

[24] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[25] Mark A Musen, Ray W Fergerson, William E Grosso, Natalya F Noy, Monica Crubezy, and John H Gennari. Component-based support for building knowledge-acquisition systems. In *Conference on intelligent information processing (IIP 2000) of the international federation for information processing world computer congress (WCC 2000)*, volume 194, 2000.

[26] Vikram Pandey. Evaluation and evolution of naonto-a profile ontology of native american diabetes patients. 2020.

[27] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[28] Eric Prud'hommeaux and Andy Seaborn. SPARQL Query Language for RDF. `https://www.w3.org/TR/sparql11-query/` [Accessed: 2023-10-20].

[29] Diogo Teles Sant'Anna, Rodrigo Oliveira Caus, Lucas dos Santos Ramos, Victor Hochgreb, and Júlio Cesar dos Reis. Generating knowledge graphs from unstructured texts: Experiences in the e-commerce field for question answering. volume 2722 of *CEUR Workshop Proceedings*, pages 56–71. CEUR-WS.org, 2020.

[30] Konstantin Schekotihin, Patrick Rodler, and Wolfgang Schmid. Ontodebug: Interactive ontology debugging plug-in for protégé. In *Foundations of Information and Knowledge Systems: 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14–18, 2018, Proceedings 10*, pages 340–359. Springer, 2018.

[31] Ming Sheng, Yuyao Shao, Yong Zhang, Chao Li, Chunxiao Xing, Han Zhang, Jingwen Wang, and Fei Gao. Dekgb: an extensible framework for health knowledge graph. In *International Conference on Smart Health*, pages 27–38. Springer, 2019.

[32] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[33] R Sivakumar and PV Arivoli. Ontology visualization protégé tools–a review. *International Journal of Advanced Information Technology (IJAIT) Vol*, 1, 2011.

[34] Serge Sonfack Sounchio, Bernard Kamsu-Foguem, and Laurent Geneste. Construction of a base ontology to represent accident expertise knowledge. *Cognition, Technology & Work*, pages 1–19, 2023.

[35] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(2):93–136, 1996.

[36] Anusha Indika Walisadeera, Athula Ginige, and Gihan Nilendra Wikramanayake. Ontology evaluation approaches: a case study from agriculture domain. In *Computational Science and Its Applications–ICCSA 2016: 16th International Conference, Beijing, China, July 4-7, 2016, Proceedings, Part IV 16*, pages 318–333. Springer, 2016.

[37] RSI Wilson, JS Goonetillake, WA Indika, and Athula Ginige. A conceptual model for ontology quality assessment. *Semantic Web*, (Preprint):1–47, 2022.

[38] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th international conference on web search and data mining*, pages 672–680, 2020.

Appendix A

Competency Question

1. CQ1:
   (a) Question: Boa tarde. Essa escova serve para o modelo ERG22? Obrigada.
   (b) Answer: Sim, esta escova serve e é compatível com erg22.
   (c) Product: 7909569301920
   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7909569301920"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Electrolux";
         UPkg:hasApplianceModel "ERG22".
}
```

2. CQ2:
   (a) Question: Serve na electrolux ll08s? caso negativo, vc teria modelo que sirva na ll08s?
   (b) Answer: Sim serve para o modelo LL08S.
   (c) Product: 7896584071887
   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7896584071887"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "electrolux";
         UPkg:hasApplianceModel "LL08S".
}
```

3. CQ3:

   (a) Question: Boa noite. Serve no Eletrolux Eco turbo?

   (b) Answer: Olá, Muito obrigado pelo interesse em nossos produtos. Sobre sua dúvida: Sim, o Conjunto Adaptador Janela Com Furo 650mm Para Ar Condicionado é da marca Electrolux e serve para complementar ou substituir o conjunto adaptador janela que acompanha o produto, adequando-se ao tamanho que você precisa.

   (c) Product: 7896584066173

   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7896584066173"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Electrolux";
         UPkg:hasApplianceModel "Eco Turbo".
}
```

4. CQ4:

   (a) Question: Serve Mr Coffee VB35 ?

   (b) Answer: Boa tarde. Resistência 127V compatível com Cafeteira Programável, BVSTDC4401, BVSTDC4401RD, BVSTDC4402.

   (c) Product: MLB1469992909

   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1469992909"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
```

```
        ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
        ?appliance UPkg:hasApplianceBrand "Mr Coffee";
             UPkg:hasApplianceModel "VB35".
    }
```

5. CQ5:

   (a) Question: Boa tarde! Esta Jarra é compatível ao Liquidificador Oster Modelo BVLB07-L00-057

   (b) Answer: Boa tarde. anunciado é compatível com o modelo informado.

   (c) Product: 7898615987624

   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
  ?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
  {?product UPkg:hasGTIN "7898615987624"}
  ?compatibility UPkg:compatibleWith ?appliance;
                rdf:type ?compatibility_type.
  ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
  ?appliance UPkg:hasApplianceBrand "Oster";
            UPkg:hasApplianceModel "BVLB07-L00-057".
}
```

6. CQ6:

   (a) Question: Boa tarde Serve panela oster modelo 8030

   (b) Answer: Olá, Infelizmente, a alça da tampa de vidro é compatível apenas com o modelo de panela elétrica Oster Gran Taste 004731R, não sendo compatível com o modelo 8030.

   (c) Product: MLB1694470731

   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1694470731"}
    ?compatibility UPkg:compatibleWith ?appliance;
                 rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Oster";
              UPkg:hasApplianceModel "8030".
}
```

7. CQ7:
   (a) Question: Boa tarde, esse copo serve para o modelo L-1000 BI 12V ?
   (b) Answer: Sim, compatível.
   (c) Product: MLB2625390411
   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2625390411"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceModel "L-1000 BI";
               UPkg:hasVolts "12V".
}
```

8. CQ8:
   (a) Question: O copo de liquidificador serve no Mondisl L 1000W ?
   (b) Answer: Sim, compatível.
   (c) Product: MLB2625390411
   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2625390411"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Mondisl";
               UPkg:hasApplianceModel "L 1000W".
}
```

9. CQ9:
   (a) Question: Ele serve no modelo special line l48
   (b) Answer: Olá, tudo bem? Não é compatível.
   (c) Product: MLB2185511558
   (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2185511558"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Mondial";
               UPkg:hasApplianceModel "L48".
}
```

10. CQ10:

    (a) Question: Serve na geladeira CRA3408ANA CONSUL 340 LITROS 110V?

    (b) Answer: Sim, serve no compressor Embraco de sua Cônsul, sim. Aguardamos sua compra conosco! À disposição.

    (c) Product: MLB1632141487

    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1632141487"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Consul";
               UPkg:hasApplianceModel "CRA3408ANA";
               UPkg:hasVolts "110V".
}
```

11. CQ11:

    (a) Question: Boa tarde serve no LG 9.000 inverter 280 v

    (b) Answer: B tarde. Para LG é outro o modelo, de tensão maior. Segue link para compra: Aguardamos sua compra conosco! À disposição.

    (c) Product: 7897738500192

    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
```

```
    {?product UPkg:hasGTIN "7897738500192"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "LG";
               UPkg:hasApplianceModel "9.000 inverter";
               UPkg:hasVolts "280v".
}
```

12. CQ12:

    (a) Question: Bom dia! Serve no compressor Embraco EM2X3134U (127V)?

    (b) Answer: B tarde. A linha EM utiliza outro modelo. Segue link para compra: Sim, temos em estoque, à pronta entrega para envio imediato. Aguardamos sua compra conosco! À disposição.

    (c) Product: 7891921307315

    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7891921307315"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Embraco";
               UPkg:hasApplianceModel "EM2X3134U";
               UPkg:hasVolts "127V".
}
```

13. CQ13:

    (a) Question: Serve para Gaggia Anima Panarello?

    (b) Answer: Boa tarde ! Sim, serve. Ficamos à disposição!

    (c) Product: 8710103073079

    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "8710103073079"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Gaggia";
               UPkg:hasApplianceModel "Anima Panarello".
}
```

14. CQ14:

    (a) Question: Boa tarde, esse interruptor serve no ASPIRADOR CYCLONE ELECTROLUX TIT10 1450 W ?.
    (b) Answer: Boa tarde! Sim, serve. Ficamos à disposição! ELETRON SERVICE.
    (c) Product: 7896347123631
    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
  ?product rdf:type UPkg:Product;
           UPkg:hasCompatibility ?compatibility.
  {?product UPkg:hasGTIN "7896347123631"}
  ?compatibility UPkg:compatibleWith ?appliance;
                 rdf:type ?compatibility_type.
  ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
  ?appliance UPkg:hasApplianceBrand "Electrolux";
             UPkg:hasApplianceModel "TIT10".
}
```

15. CQ15:

    (a) Question: Da serto no processador Philips série 3000 Potencoa 600w Ri7300?
    (b) Answer: Olá ! Sim, serve. Qualquer outra dúvida, basta nos contatar novamente!.
    (c) Product: 8710103054665
    (d) SPARQL:

```
select ?product ?compatibility_type ?appliance
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "8710103054665"}
    ?compatibility UPkg:compatibleWith ?appliance;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?appliance UPkg:hasApplianceBrand "Philips";
               UPkg:hasApplianceModel "Série 3000 Potencoa 600w Ri7300".
}
```

16. CQ16:

   (a) Question: Serve na KIA BONGO 20/21?
   (b) Answer: Boa tarde , serve sim . ZND PEÇAS Agradece o contato e aguarda sua compra
   (c) Product: MLB2719751431
   (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2719751431"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "kia";
         UPkg:hasCarModel "Bongo";
         UPkg:hasCarYear "20/21".
}
```

17. CQ17:

   (a) Question: Ela serve para bomho k2700 4x4 ano 2009?
   (b) Answer: Serve sim
   (c) Product: MLB956925197
   (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB956925197"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "bomho";
         UPkg:hasCarModel "K2700";
         UPkg:hasCarYear "2009".
}
```

18. CQ18:

   (a) Question: serve na HYUNDAI HR 2010?
   (b) Answer: Boa tarde! esse não serve, segue o link correto, marca japonesa com qualidade superior a cofap.

(c) Product: MLB3387033577

(d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB3387033577"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "hyundai";
         UPkg:hasCarModel "HR";
         UPkg:hasCarYear "2010".
}
```

19. CQ19:

    (a) Question: Desculpa. . . não sei se corresponde. Seria para o pneu fino de estepe do HB20 2019, no pneu diz q a velocidade máxima é de 80 km.

    (b) Answer: É adequado para o uso em um HB20 2019.

    (c) Product: MLB1717002883

    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1717002883"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "Hyundai";
         UPkg:hasCarModel "HB20";
         UPkg:hasCarYear "2019".
}
```

20. CQ20:

    (a) Question: Boa tarde Esse pneu pode ser utilizado na Santa Fé 2008 Os que eu tenho na Santa Fé são 235 60 18.

    (b) Answer: Boa tarde. Se as medidas do produto correspondem às utilizadas atualmente deverá atender sim.

    (c) Product: MLB2761007508

    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2761007508"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarModel "Santa Fé";
         UPkg:hasCarYear "2008".
}
```

21. CQ21:
    (a) Question: Boa tarde. Serve na Frontier Attack 2019?
    (b) Answer: Boa tarde. Se as medidas do produto correspondem às utilizadas atualmente deverá atender sim.
    (c) Product: MLB1771694609
    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1771694609"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "nissan";
         UPkg:hasCarModel "Frontier";
         UPkg:hasCarYear "2019".
}
```

22. CQ22:
    (a) Question: Serve no Siena fire 1.4 8v 2015? Obs não é motor evo.
    (b) Answer: Olá! Tudo bem? Não serve.
    (c) Product: MLB1131399755
    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1131399755"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "fiat";
         UPkg:hasCarModel "Siena";
         UPkg:hasCarYear "2015".
}
```

23. CQ23:

    (a) Question: serve na palio weekend adventure 1.6 16v ano 2002/03? grato.

    (b) Answer: Olá! Tudo bem? Serve sim.

    (c) Product: 7897483464145

    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "7897483464145"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "fiat";
         UPkg:hasCarModel "Palio Weekend Adventure";
         UPkg:hasCarYear "2002/03".
}
```

24. CQ24:

    (a) Question: Serve na ranger 2008 2.3 gasolina ?

    (b) Answer: Olá! Tudo bem? Sim, serve na Ranger 2008 2.3 gasolina.

    (c) Product: MLB1148231448

    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1148231448"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "Ford";
         UPkg:hasCarModel "Ranger";
```

```
          UPkg:hasCarYear "2008".
}
```

25. CQ25:

    (a) Question: 9bwab45u4kt047833 serve para este gol 2018/19. 1.6 msi
    (b) Answer: Bom dia, infelizmente nao da certo e nao temos a peça correta.
    (c) Product: MLB1670791125
    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1670791125"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "VW";
         UPkg:hasCarModel "Gol";
         UPkg:hasCarYear "2018/19".
}
```

26. CQ26:

    (a) Question: Esse comando serve no gol g5 2010 1.0 chassi 9BWAAO5UOBT136830
    (b) Answer: Boa noite! Aplica perfeitamente em seu veículo.
    (c) Product: MLB2730966123
    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
         UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2730966123"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarModel "Gol";
         UPkg:hasCarYear "2010".
}
```

27. CQ27:

    (a) Question: Serve na Titan 2007 es?

   (b) Answer: Serve sim amigo perfeitamente produto na medida correta para sua moto!!
   (c) Product: MLB1446591954
   (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB1446591954"}
    ?compatibility UPkg:compatibleWith ?car;
                  rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarBrand "honda";
         UPkg:hasCarModel "Titan";
         UPkg:hasCarYear "2007".
}
```

28. CQ28:
    (a) Question: Essa peça da certo na Titan 150 /2007
    (b) Answer: Boa tarde tudo bom ? Serve perfeitamente! Produto de ótima qualidade, nota fiscal e garantia!
    (c) Product: MLB2016966667
    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2016966667"}
    ?compatibility UPkg:compatibleWith ?car;
                  rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarModel "Titan 150";
         UPkg:hasCarYear "2007".
}
```

29. CQ29:
    (a) Question: Serve na g310 2021?
    (b) Answer: Boa tarde tudo bom ? Serve perfeitamente! Produto de otima qualidade, nota fiscal e garantia!
    (c) Product: MLB2750979998

(d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2750979998"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarModel "g310";
         UPkg:hasCarYear "2021".
}
```

30. CQ30:

    (a) Question: Esse quite serve pra Twister 2021

    (b) Answer: Bom dia. Negativo. Serve para Todas CB 300 Apenas.

    (c) Product: MLB2794025012

    (d) SPARQL:

```
select ?product ?compatibility_type ?car
where {
?product rdf:type UPkg:Product;
          UPkg:hasCompatibility ?compatibility.
    {?product UPkg:hasGTIN "MLB2794025012"}
    ?compatibility UPkg:compatibleWith ?car;
                   rdf:type ?compatibility_type.
    ?compatibility_type rdfs:subClassOf UPkg:Compatibility.
    ?car UPkg:hasCarModel "Twister";
         UPkg:hasCarYear "2021".
}
```

Appendix B

KEOD Copyright Acceptance

Gmail **Jesamin Zevallos <jesaminzev@gmail.com>**

Authorization to include the paper

2 mensajes

Jesamin Zevallos <jesaminzev@gmail.com> 8 de noviembre de 2023, 21:50
Para: keod.secretariat@insticc.org

Dear editors of KEOD,
I'm writing this email because I need your authorization to include the paper "UpKG: A Framework to Insert New Domains in Knowledge Graphs" in my master dissertation, supervised by my advisor Prof Dr. Julio Cesar dos Reis.
The rules of my institution (Unicamp) states that: "In case of a published paper, the student should collect the editorial board/committee authorization to include it in his/her dissertation/thesis."
The mentioned paper is a key part of the whole dissertation and without it, the work would be incomplete. Could you provide this in the next few days?

Best regards,
Jesamin Zevallos.

KEOD Secretariat <KEOD.Secretariat@insticc.org> 9 de noviembre de 2023, 06:54
Responder a: KEOD.Secretariat@insticc.org
Para: jesaminzev@gmail.com

Dear Jesamin Zevallos,

Thank you for your e-mail. You can include the KEOD paper in your thesis, as long as you include a reference where it was previously published. You can use any version, but we prefer that the published version included in the Proceedings is the one used.

--
Best regards,
Ana Rita Paciência
KEOD Secretariat

INSTICC Office
Avenida de S. Francisco Xavier, Lote 7 Cv. C,
2900-616 Setubal, Portugal
Tel: +351 265 520 185
Fax: +351 265 520 186
http://www.keod.scitevents.org/

On Thursday, November 9th 2023, 1:50 am CET (+0100), Jesamin Zevallos wrote:

> Dear editors of KEOD,
> I'm writing this email because I need your authorization to include the paper "UpKG: A Framework to Insert New Domains in Knowledge Graphs" in my master dissertation, supervised by my advisor Prof Dr. Julio Cesar dos Reis.
> The rules of my institution (Unicamp) states that: "In case of a published paper, the student should collect the editorial board/committee authorization to include it in his/her dissertation/thesis."
> The mentioned paper is a key part of the whole dissertation and without it, the work would be incomplete. Could you provide this in the next few days?
>
> Best regards,
> Jesamin Zevallos.