


# UpKG: A Framework to Insert New Domains in Knowledge Graphs

Jesamin Melissa Zevallos-Quispe<sup>1</sup><sup>a</sup>, André Gomes Regino<sup>1</sup><sup>b</sup>, Víctor Jesús Sotelo Chico<sup>1</sup><sup>c</sup>,  
Víctor Hochgreb de Freitas<sup>2</sup><sup>d</sup> and Julio Cesar dos Reis<sup>1</sup><sup>e</sup>

<sup>1</sup>*Institute of Computing, University of Campinas, Campinas, Brazil*

<sup>2</sup>*GoBots, São Paulo, Brazil*

**Keywords:** Knowledge Graphs, Domain Extension, e-Commerce, Compatibility, Ontology, SPARQL.

**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 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. This study 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 in the real-world context of the GoBots company to facilitate the new domain insertion process. Our framework was applied in GoBots company, specialized in e-commerce solutions in Latin America. The conducted case was on an existing KG, whose triples were dominated by the Automobile domain. Our work covered and inserted novel triples concerning the Appliances domain. The conducted evaluation shows that we can verify the feasibility and applicability of the UpKG framework.

## 1 INTRODUCTION


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 (Li et al., 2020) (Sant’Anna et al., 2020) (Chen et al., 2019) because several applications related to online commerce explore them in different areas, such as product searches, product recommendations, and virtual chats, among others. Several big-player companies have used KGs in e-commerce, including Alibaba (Li et al., 2020), Walmart (Xu et al., 2020), and Farfetch (Barroca et al., 2022), 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, keeping a KG updated involves expanding it and inserting domains such as appliances, furniture, and automobiles.


In addition, inserting new domains into a KG is not trivial because the ontology expert needs to analyze and understand the components, such as ontology structure, relations, instances, SPARQL queries, and data insertion to the KG with the new domain, among other aspects. For example, suppose an online store has a KG of only Furniture items, and a customer requests information on a product from the Clothing domain. In that case, this KG cannot return an adequate response to the customer. For this reason, inserting the ‘Clothing’ domain in such KG leads to a better quality of response and greater customer satisfaction.


The literature presents research studies related to KGs in e-commerce (Li et al., 2020) (Chen et al., 2019) (Sant’Anna et al., 2020). Some studies explore the extension of domains in KGs (Ait-Mlouk and Jiang, 2020; Sheng et al., 2019) and ways of populating triples in KGs (Kertkeidkachorn and Ichise, 2018) (Sant’Anna et al., 2020). However, none of the

<sup>a</sup> <https://orcid.org/0009-0006-6419-4694>

<sup>b</sup> <https://orcid.org/0000-0001-9814-1482>

<sup>c</sup> <https://orcid.org/0000-0001-9245-8753>

<sup>d</sup> <https://orcid.org/0000-0002-0529-7312>

<sup>e</sup> <https://orcid.org/0000-0002-9545-2098>

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 the KG.

In this study, we propose the **UpKG** framework, which allows inserting new domains in an existing KG with an e-commerce approach based on user questions and answers. Our study presents the following contributions:

- Conceptually develop UpKG focusing on e-commerce context based on user questions and answers.
- Evaluate the proposed UpKG framework in the real-world context of an AI startup in Latin America.
- Ensure that our framework is flexible enough to add new domains focusing on e-commerce.

Our evaluation was conducted in the context of the GoBots<sup>1</sup> company, where we inserted triples of the Appliances domain into a KG composed of triples of the Automobiles domain. It was possible to insert 260 individuals for both appliances and automobiles. In addition, our results showed that it is possible to answer questions from users with SPARQL queries relying on the new version obtained for the KG. We verified the ontology's consistency and coherence with the OntoDebug<sup>2</sup> plugin of Protégé. The main goal of *OntoDebug* is to help users find those axioms in ontologies, giving information if an ontology is inconsistent.

The remainder of this article is organized as follows. Section 2 presents a summary of related work. Section 3 describes the UpKG framework. Section 4 presents the evaluation of our framework, where we show the context and the results obtained from applying our framework in an existing real-world setting. Section 5 discusses our findings. Finally, Section 6 concludes the study and describes future work.

## 2 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.** Li *et al.* (Li et al., 2020) 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.* (Li et al., 2020) described how the domain KG comprises free text and tested it with various applications in e-commerce categories.

Xu *et al.* (Xu et al., 2020) proposes 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, since it is focused on products and how their information can be used for recommendation and classification tasks.

**Populating KG.** The T2KG (Kertkeidkachorn and Ichise, 2018) 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 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.* (Sant'Anna et al., 2020) 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.* (Sant'Anna et al., 2020) focused on the e-commerce domain.

Integroly (Guedea-Noriega and García-Sánchez, 2022) 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

<sup>1</sup>Company specialized in artificial intelligence solutions for e-commerce platforms in Latin America. Site available in <https://gobots.ai/>

<sup>2</sup><http://isbi.aau.at/ontodebug/>

Table 1: Comparison in three scopes of studies in KG connected to our present investigation in this article.

|   | <b>KG in e-commerce</b> | <b>Extending KG</b> | <b>Populating KG</b> |
|---|-------------------------|---------------------|----------------------|
| (Kertkeidkachorn and Ichise, 2018)        |                         |                     | X                    |
| (Sheng et al., 2019)                      |                         | X                   |                      |
| (Xu et al., 2020)                         | X                       |                     |                      |
| (Ait-Mlouk and Jiang, 2020)               |                         | X                   |                      |
| (Li et al., 2020)                         | X                       |                     |                      |
| (Sant’Anna et al., 2020)                  | X                       |                     | X                    |
| (Guedea-Noriega and García-Sánchez, 2022) |                         |                     | X                    |
| <b>Our proposal</b>                       | <b>X</b>                | <b>X</b>            | <b>X</b>             |

news. They found problems with synonyms, ambiguities, and the connections made in the KG.

**Extending KG.** KBot (Ait-Mlouk and Jiang, 2020) 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 (Sheng et al., 2019) 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 1 shows the organization and conceptual analysis we performed for the existing literature reviewed. We considered three scopes.

Our study aims to cover these three aspects: (i) KG in e-commerce, use of KG in an e-commerce platform; (ii) Populating KG, populate a KG with new knowledge based on the ontology; and (iii) Extending KG, modify 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. The following section describes our framework in detail.

### 3 FRAMEWORK UpKG

We propose a framework to insert new domains into an existing KG focused on e-commerce from questions and answers. Figure 1 presents the general pipeline of our framework. It is composed of four

modules: (i) Understand the current KG, (ii) Identify a new domain, (iii) Restructure the ontology, (iv) Populate the KG. These modules are further elaborated in the following.

#### 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. 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  $v_0$ . This step can be performed by a person with knowledge of ontologies, observing the properties and attributes of the data. 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, you can use some tools to understand the ontology structure; one of the best-known is Protégé. 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 (answer users’ questions). The fourth step is to identify the domains currently supported by the KG  $v_0$  and thus map the domains that are necessary to insert. In this step, the human person can access the current KG information and discern which domains the KG covers.

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

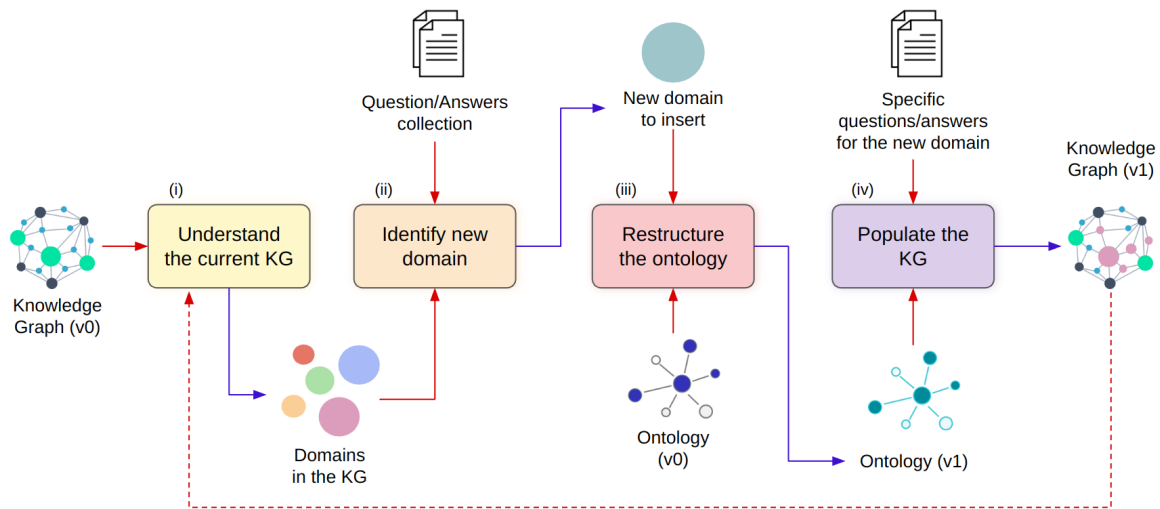


Figure 1: Overview of the UpKG Framework. (i) Understand the current KG: Module focused on understanding the current KG and its base ontology; (ii) Identify new domain: Module focused on discovering the new domain that will be inserted; (iii) Restructure the ontology: Critical module in which the base ontology is combined with the new properties; (iv) Populate the KG: Final module focused on generating the triples to insert the KG.

### 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. The first step is to identify all the compatibility questions not answered by the existing KG. This step can be done automatically through a script, which has as its input a collection of user questions and answers. The saved questions are those that have no answer from the KG. A person with programming knowledge can execute this. The second step is identifying relationships that the ontology  $v_0$  cannot map, making it impossible to respond to the KG according to its intent. The third step refers to identifying the domain with the highest number of unanswered questions from the current KG. To find the domain to insert, you can create a script that generates a histogram automatically, which represents the domain and its number of unanswered questions. This allows a graphical display of domains that do not cover the current KG; we can decide which domain to include with this histogram.

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

### 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 3.2. The input

data to be used in this module is the new domain to be inserted and the existing ontology denoted as  $v_0$ ; usually, an ontology is modified based on the new needs of a system. Therefore, it is essential to modify the structure of knowledge. The data from this module is the ontology with the new domain to which we denote it by the version  $v_1$ .

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 review of the literature. The second step synthesizes the ontologies with a set of properties already used in other contexts, which can help us in our current KG and thus start from a base ontology. The third step is to obtain the new ontology properties necessary for the new domain and the type of questions to ask; for example, if our current domain is Automobiles, we could not answer questions that refer to Technology. In this step, the intervention of human reasoning 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, by analyzing the structure of questions. The fourth step manually combines the properties that were previously mapped with our ontology  $v_0$ ; in this step, we can use a tool that facilitates ontology modeling (eg. Protégé<sup>3</sup>) to obtain our ontology  $v_1$  as output.

This module can be developed manually because

<sup>3</sup><https://protege.stanford.edu/>

in all the steps human intervention is necessary to re-structure the ontology.

### 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  $v_1$  to be added to the KG.

The first step is to obtain the information for each question and answer from a specific domain, which will be collected from the e-commerce system in which this framework will be applied. 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. This process is based on the work of Sant’Anna *et al.* (Sant’Anna *et al.*, 2020), who proposed their own Natural Language Understanding (NLU) process, where they defined the concepts of entity and intent.

Sant’Anna *et al.* (Sant’Anna *et al.*, 2020) defined 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.

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. The fourth step is to store the collection of triplets in a database, this can be local or remote. This step aims to access information through SPARQL queries. In this step you can use tools that store RDF data, as well as GraphDB<sup>4</sup>, Virtuoso<sup>5</sup>, among others.

Finally, the output data of this last step is the KG  $v_1$ . This module can be developed automatically because each step can be executed through a script.

## 4 EVALUATION

Our framework was carried out in the context of the GoBots and offers solutions to structure the knowledge of compatibility between products in a KG based on an ontology. The company offers a solution of questions and answers for e-commerce platforms. We describe the results of applying the process of each framework module in the following.

### 4.1 Results Applying the upKG Framework

#### 4.1.1 Results in Understanding the Current KG

Figure 2 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  $v_0$ . First, 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.

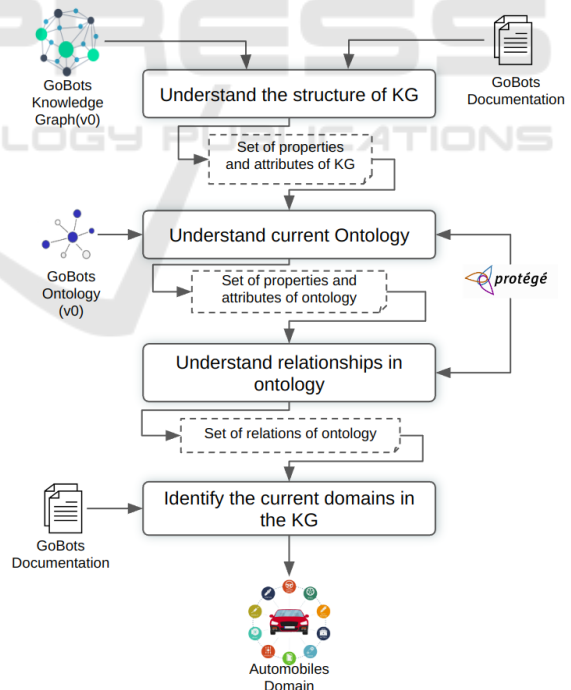


Figure 2: Results in understanding the current KG in GoBots.

As a result of understanding the structure of KG, we have a set of properties and attributes of the KG  $v_0$ . Figure 3 shows the initial ontology, in which we

<sup>4</sup><https://www.ontotext.com/products/graphdb/>

<sup>5</sup><https://virtuoso.openlinksw.com/>



observe the existing classes, properties, and attributes to store the compatibility knowledge between a product and a Car type object. In the second step, we used the initial ontology  $v_0$  proposed in (Sant'Anna et al., 2020). 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 have a set of properties and attributes in the ontology  $v_0$ .

In the third step, 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  $v_0$ .

In the fourth step, we defined the domain or domains that cover the current KG. In this case, the current domain is *Automobile*, 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*.

#### 4.1.2 Result in Identifying New Domain

GoBots collected a set of pairs of questions and answers related to a product that human attendants answered. This collection is stored in a MongoDB database. Figure 4 shows the process of identifying the new domain to be inserted in the new version of the KG. For the first step, we identified all the questions that the current KG cannot answer, so we do not consider the questions related to the domain of Automobiles. 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. In this experiment, the date range was two months. As a result of the first step is a collection of questions that the KG did not answer.

In the second step, we identified 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  $v_0$ . 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  $v_0$ . As a result of the second step, we have a set of relationships that were not

mapped by the current ontology.

In the third step, we created a script in Python to identify the domains (categories) with the largest number of questions sorted descending; the three domains with the most questions and answers were Appliances, Home and Furniture, and Tools. In the next steps, we present the insertion of the domain Appliances to the new version of the ontology and later to KG.

#### 4.1.3 Result in Restructuring the Ontology

The ontology  $v_0$  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. The ontology obtained from applying our framework was denoted as  $v_1$ , which can cover two domains of questions users ask in an e-commerce system.

Figure 5 shows the process of restructuring the GoBots ontology to accept the Appliance domain. The first step is to perform an ontology search; in our case, they are appliances. 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*<sup>6</sup>, which provides a list of properties in *Product* and allows us to identify the properties needed for the new domain. For this reason, in the second step, we selected *Schema.org* as the base for our ontology.

The third step is to identify the new properties necessary for our ontology. In this case, we used standard properties present in *Schema.org* regarding *Product*, such as *Model*, *Brand*, and *Material*. We also created new properties that were not found in *Schema.org*. These are created from human reasoning to map the relationships presented in the questions about Appliances: *Voltage* and *Volume*.

The fourth step was to combine the properties; in this step, we used the *Protégé tool* to combine our property list with the existing GoBots ontology  $v_0$ . The final result of this step is an ontology that accepts two domains: Automobiles and Appliances. We denoted this ontology as  $v_1$ .

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: *hasVolume* and *hasVolts*.

Figure 9 shows in Protégé tool visualization all

<sup>6</sup><https://schema.org/Product>

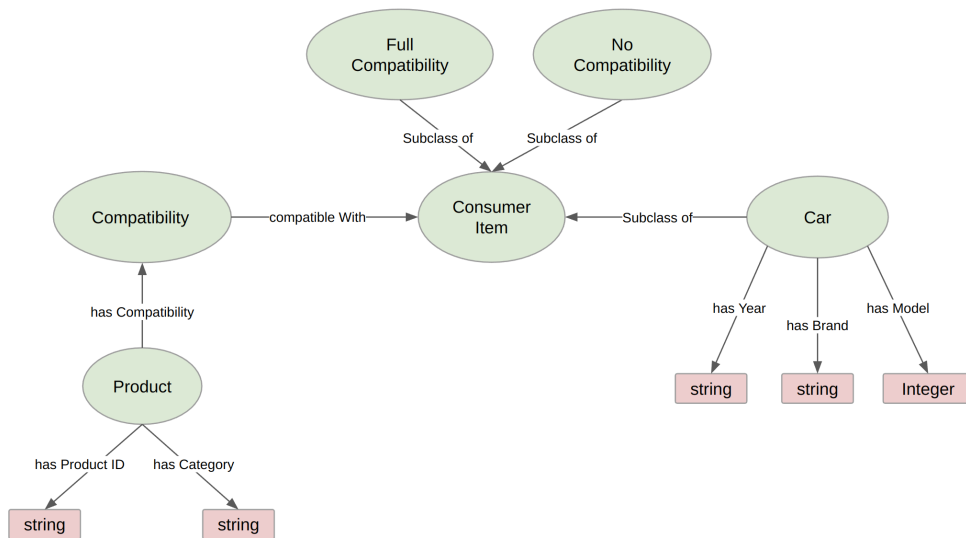


Figure 3: Ontology v0 that represents the compatibility knowledge between a product (a car seatbelt, for instance) and a consumer item (an Audi TT, for instance) (Sant’Anna et al., 2020).

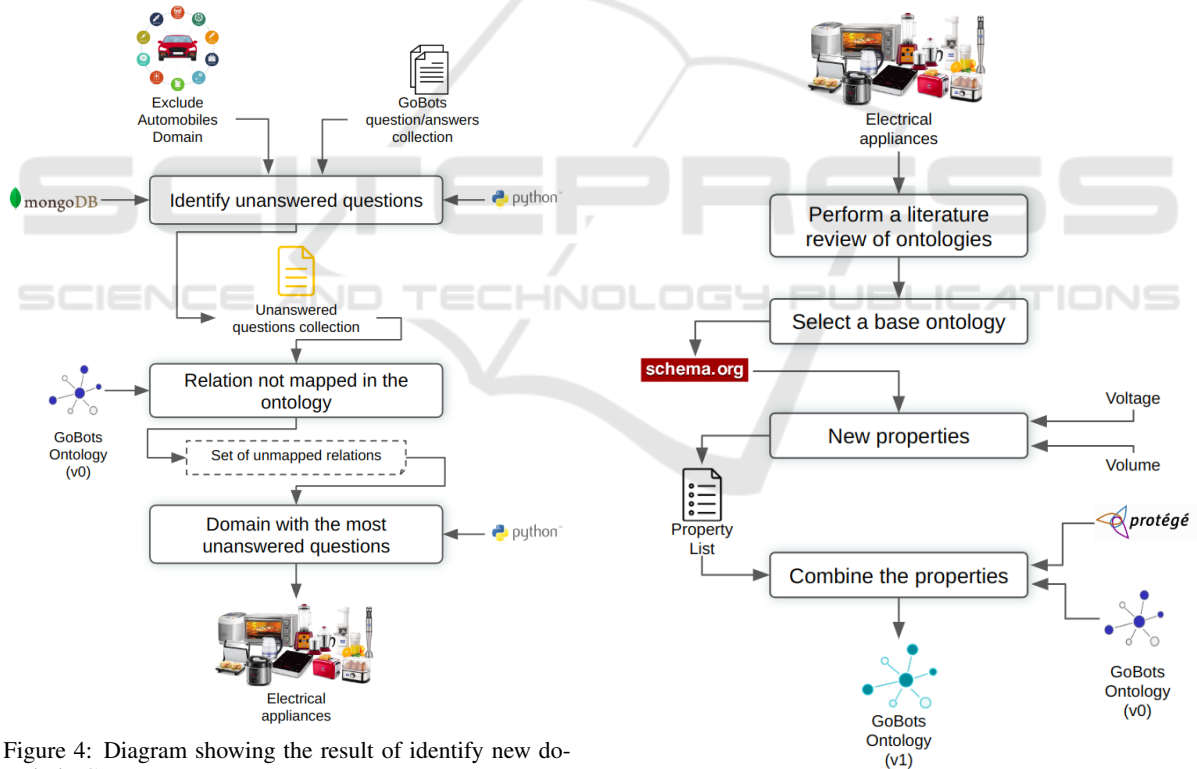


Figure 4: Diagram showing the result of identify new domain in GoBots.

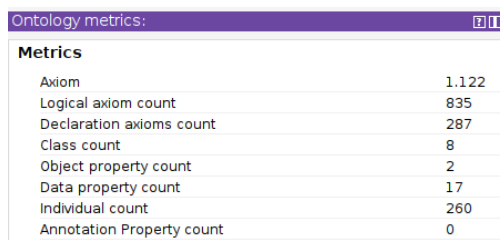
Figure 5: Results in restructuring the GoBots ontology with Appliance Domain.

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 (Sant’Anna et al., 2020). 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 6 shows the metrics of the ontology struc-

ture in terms of classes, properties, and axioms. The ontology  $v_0$  presents 7 classes while our ontology  $v_1$ . Figure 6 presents a total of 8 classes. This shows that inserting a new domain in ontology was simple, considering that ontology  $v_0$  was well-designed to accept other domains.



| Ontology metrics:         |       |
|---------------------------|-------|
| Metrics                   |       |
| Axiom                     | 1.122 |
| Logical axiom count       | 835   |
| Declaration axioms count  | 287   |
| Class count               | 8     |
| Object property count     | 2     |
| Data property count       | 17    |
| Individual count          | 260   |
| Annotation Property count | 0     |

Figure 6: Metrics of the ontology  $v_1$ .

Figure 7 graphically represents how the  $v_1$  ontology was structured.

Figure 8 and Figure 9 show the Object Property and Data Property of the ontology  $v_1$ , respectively.

The structure and relationships of the ontology  $v_1$  were validated using Protégé built-in debugging tool OntoDebug<sup>7</sup>. OntoDebug (Schekotihin et al., 2018) is a Protégé plugin that implements an interactive approach to ontology debugging, since having a faulty ontology the tool finds a set of faulty axioms that explain the problem. The main goal of OntoDebug is to help find those axioms in ontologies, giving information if an ontology is inconsistent. Figure 10 presents our final ontology  $v_1$  validated with OntoDebug. We obtained as a result of the execution, that our ontology is coherent and consistent.

#### 4.1.4 Result in Populating the KG

To populate our KG, we collected a set of questions and answers from the GoBots database between April 01, 2023 and June 01, 2023, obtaining a total of 47142 questions. We filtered out those with Compatibility intent, obtaining 137 questions.

Figure 11 shows the automatic process of populating the KG. The first step is to obtain the product information related to the question. This extraction of information was carried out by a POST request, which returns the necessary attributes (SKU, GTIN or ProductID) to create a product in our ontology  $v_1$  (cf. Figure 7).

The second step was the extraction of the intents and entities of the questions and answers collected from the GoBots database. This collection of questions and answers is related to the domain of Appliances. In this step, we defined and created an original

Python Script to identify the questions with Compatibility intent.

Once we have filtered the questions with the compatibility intent, we proceed with the automatic extraction of the entities involved. As an example, the question *"Is it compatible with LSE09 220v?"*, from which we extract the entities present, such as the Model and Voltage. To perform this extraction, we use a script, which receives a question as input and returns all the entities defined (domain, model, brand, volume, volts, design, and material).

As a result of this extraction, we have an intents/entities collection. The third step consists of the creation of RDF triplets; it is necessary to emphasize that this step is performed automatically, like the previous two; this step is responsible for generating all the relationships allowed by our ontology  $v_1$  for the different domains, among them generating instances of Automobiles and Appliances by triplets. These are possible to make thanks to the collection of extracted entities.

The fourth step was to store the data in the RDF Stores. At this stage, we explored GraphDB<sup>8</sup>, which is a database that stores graphs. This tool was used to store the collected triples and execute SPARQL queries, thus populating the new KG  $v_1$  with questions and answers from the Appliances and Automobiles domain.

Of these 137 questions, the automatic insertion was done directly with a script to the OWL file generated by Protégé. However, not all the questions were inserted, as some did not have all the necessary entities to generate knowledge. A total of 260 individuals were automatically inserted: 53 individuals in the Automobiles domain, 34 individuals in the Appliances domain, 84 individuals in Product, 85 in FullCompatibility, and finally, 4 in NoCompatibility, this being our KG  $v_1$ .

## 4.2 Results in Using the Outcome KG

With our KG  $v_1$  populated, we access the data through SPARQL queries. These queries were defined for different relevant user questions, taking into account the Compatibility intent.

To present the generated KG's value, quality, and effectiveness, we show questions for the "Automobiles" and "Appliances" domains with their respective SPARQL queries.

- *"Is it compatible with LSE09 220v?"*. This question refers to the product with GTIN 763293865672, where they are asking if the prod-

<sup>7</sup><http://isbi.aau.at/ontodebug/>

<sup>8</sup><https://www.ontotext.com/products/graphdb/>



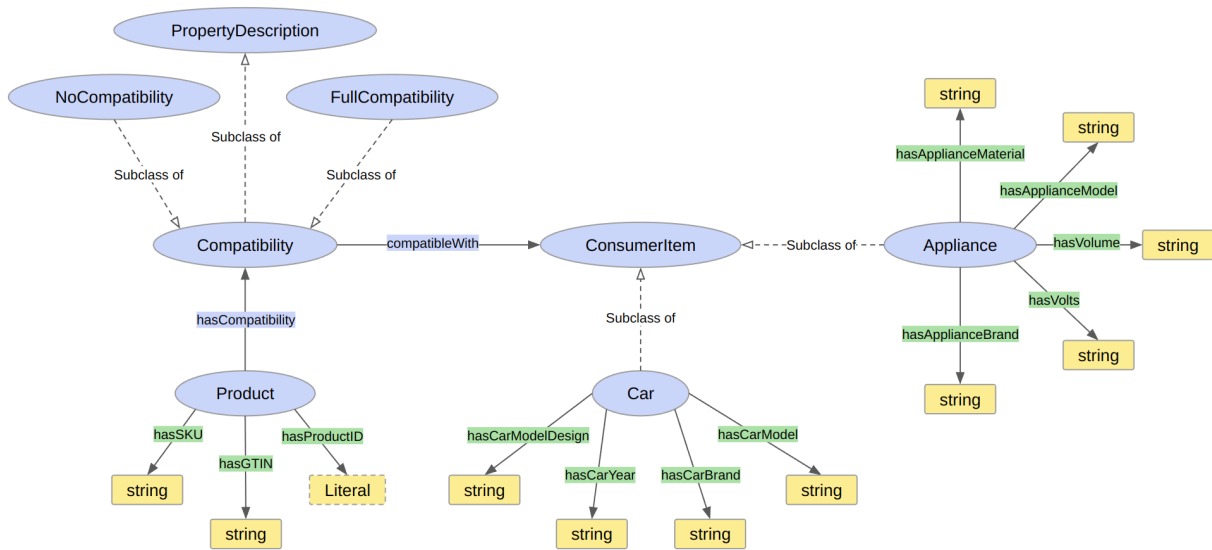


Figure 7: Ontology structure v1. Represents the compatibility between products and an object of Car or Appliance type.

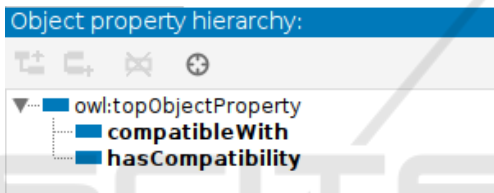


Figure 8: Object Property of the ontology v1.

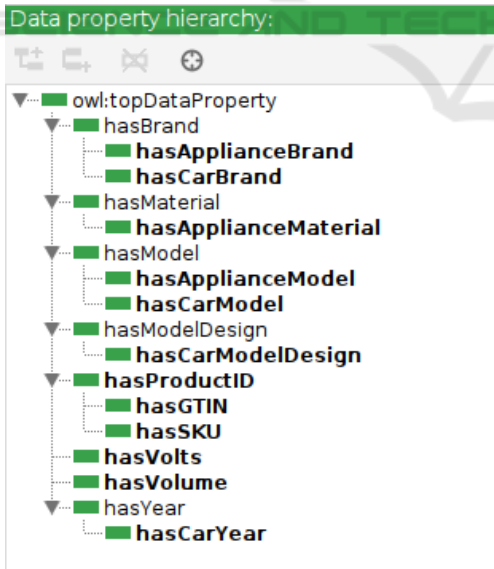


Figure 9: Data Property of the ontology v1.

uct is compatible with a LSE09 model washing machine (Appliance) and that it is compatible with a 220V voltage. The query generated in SPARQL would be the following:

```
select ?product ?comp_type ?appliance
where {
  ?product rdf:type UPkg:Product;
    UPkg:hasCompatibility ?comp.
  {?product UPkg:hasGTIN "763293865672"}
  ?comp
    UPkg:compatibleWith ?appliance;
    rdf:type ?comp_type.
  ?comp_type
    rdfs:subClassOf UPkg:Compatibility.
  ?appliance
    UPkg:hasApplianceModel "LSE09";
    UPkg:hasVolts "220V".
}
```

• *Is this pump compatible with Ford fiesta Zetec year 2003?*. This question refers to the product with GTIN 827365281647, if it is compatible with a Ford (Car), model Fiesta and year 2003. The query generated in SPARQL would be the following:

```
select ?product ?comp_type ?car
where {
  ?product rdf:type UPkg:Product;
    UPkg:hasCompatibility ?comp.
  {?product UPkg:hasGTIN "827365281647"}
  ?comp
    UPkg:compatibleWith ?car;
    rdf:type ?comp_type.
  ?comp_type
    rdfs:subClassOf UPkg:Compatibility.
  ?car
    UPkg:hasCarBrand "Ford";
    UPkg:hasCarModel "Fiesta";
    UPkg:hasYear "2003".
}
```

• *Good morning, my washing machine is a*

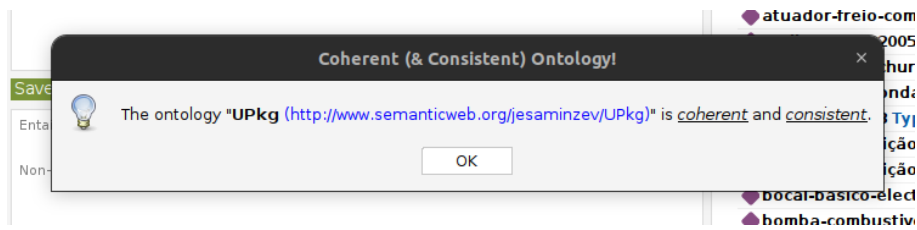


Figure 10: The debug output of the UpKG ontology shows that it is consistent and coherent.

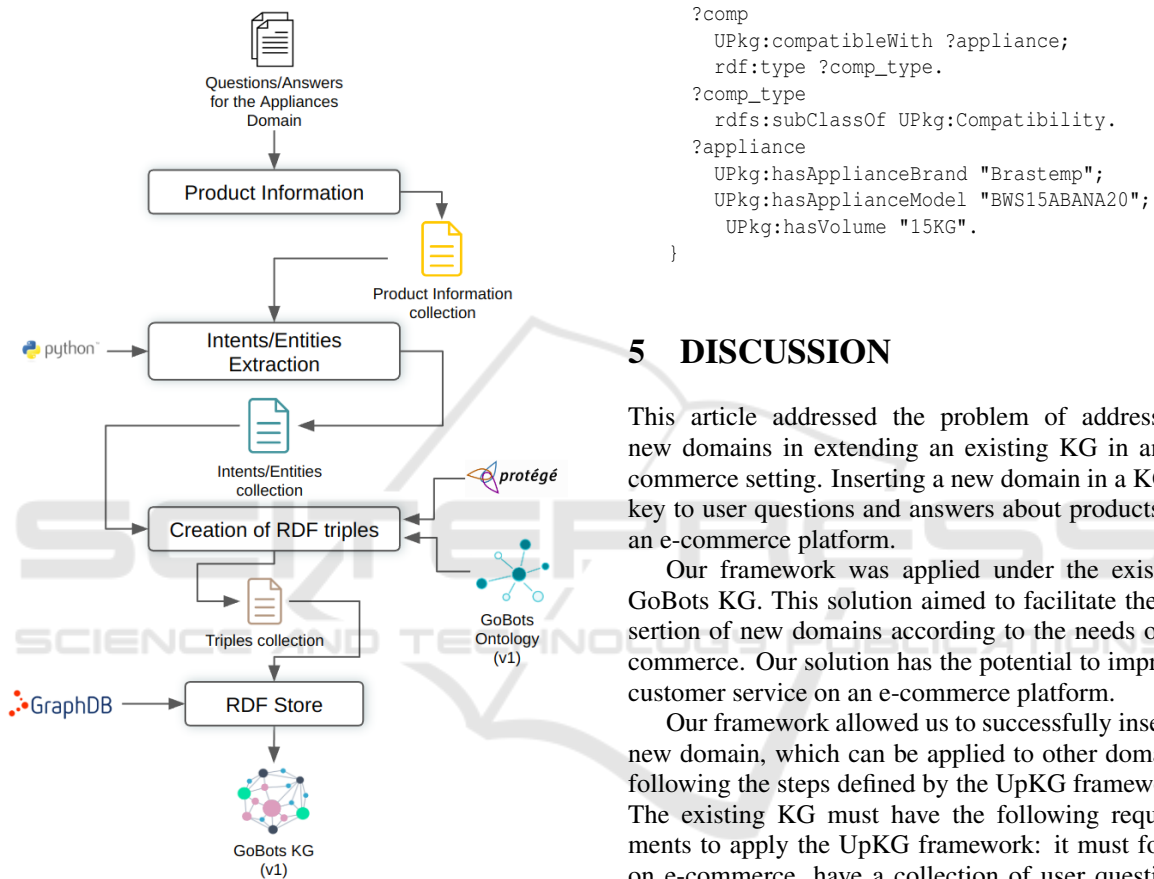


Figure 11: Results in populating the KG with Appliance Domain.

*Brastemp 15 KG and model BWS15ABANA20, Is this valve model compatible?.* This question refers to the product with GTIN 1234567890123, they are asking if the product is compatible with Brastemp brand, BWS15ABANA20 model and 15KG of volume washing machine (Appliance). The query generated in SPARQL would be the following:

```
select ?product ?comp_type ?appliance
where {
  ?product rdf:type UPkg:Product;
  UPkg:hasCompatibility ?comp.
  {?product UPkg:hasGTIN "1234567890123"}
```

```
?comp
  UPkg:compatibleWith ?appliance;
  rdf:type ?comp_type.
?comp_type
  rdfs:subClassOf UPkg:Compatibility.
?appliance
  UPkg:hasApplianceBrand "Brastemp";
  UPkg:hasApplianceModel "BWS15ABANA20";
  UPkg:hasVolume "15KG".
}
```

## 5 DISCUSSION

This article addressed the problem of addressing new domains in extending an existing KG in an e-commerce setting. Inserting a new domain in a KG is key to user questions and answers about products on an e-commerce platform.

Our framework was applied under the existing GoBots KG. This solution aimed to facilitate the insertion of new domains according to the needs of e-commerce. Our solution has the potential to improve customer service on an e-commerce platform.

Our framework allowed us to successfully insert a new domain, which can be applied to other domains following the steps defined by the UpKG framework. The existing KG must have the following requirements to apply the UpKG framework: it must focus on e-commerce, have a collection of user questions and answers, and have questions answered by human attendees.

As we observed results in Section 4.2, the application of our framework enabled the insertion of a new domain to an existing KG in a real-world environment. We reached a new version of an ontology considering classes that represent knowledge in both domains. We addressed the objects that link our classes and, finally, the properties of the classes, respectively. In the same way, it is important to have many relationships between a product and an object of type *ConsumerItem* to be able to answer several questions from consumers. From another perspective, using all our stored knowledge for different tasks, including product recommendations, is possible.

The application of the UpKG framework was de-

veloped using real GoBots data relying on a collection of questions and answers. The stage of populating the KG appeared to be the most complicated because you need to automate various processes, including Extracting question intents/entities, creating triples, and storing the data in an RDF Store. However, automating this popular KG process allows us to process more data in less time than a manual process. Furthermore, it allowed us to evaluate our UpKG framework completely.

## 6 CONCLUSION

Inserting new domains in an existing KG focused on e-commerce is an absolute necessity to enable covering a large amount of knowledge because it is constantly under evolution. This study proposed a framework suited to inserting new domains in a KG. Four main modules were proposed: Understanding the current KG, Identifying new domains, Restructuring the ontology, and Populating the KG. Our framework was applied in the context of a real-world company using a collection of questions and answers about Appliances. We showed that the framework application was developed considering the addition of new individuals, obtaining a coherent and consistent KG within the new domain, evidencing the viability and applicability of the framework. In future work, we plan to bring the KG v1 to production to have a further consistent evaluation assessing questions asked by users in real time. Finally, we aim to insert new domains detected in stage two of our framework to have more knowledge in our KG.

## ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001 and by the São Paulo Research Foundation (FAPESP) (Grant #2022/13694-0)<sup>9</sup>. We also would like to thank GoBots for collecting data and sharing their environment.

## REFERENCES

Ait-Mlouk, A. and Jiang, L. (2020). Kbot: a knowledge graph based chatbot for natural language understanding over linked data. *IEEE Access*, 8:149220–149230.

- Barroca, J., Shivkumar, A., Ferreira, B. Q., Sherkhonov, E., and Faria, J. (2022). Enriching a fashion knowledge graph from product textual descriptions. *arXiv preprint arXiv:2206.01087*.
- Chen, S., Li, C., Ji, F., Zhou, W., and Chen, H. (2019). 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.
- Guedea-Noriega, H. H. and García-Sánchez, F. (2022). Integroly: Automatic knowledge graph population from social big data in the political marketing domain. *Applied Sciences*, 12(16):8116.
- Kertkeidkachorn, N. and Ichise, R. (2018). An automatic knowledge graph creation framework from natural language text. *IEICE TRANSACTIONS on Information and Systems*, 101(1):90–98.
- Li, F.-L., Chen, H., Xu, G., Qiu, T., Ji, F., Zhang, J., and Chen, H. (2020). 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.
- Sant’Anna, D. T., Caus, R. O., dos Santos Ramos, L., Hochgreb, V., and dos Reis, J. C. (2020). 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.
- Schekotihin, K., Rodler, P., and Schmid, W. (2018). 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.
- Sheng, M., Shao, Y., Zhang, Y., Li, C., Xing, C., Zhang, H., Wang, J., and Gao, F. (2019). Dekgb: an extensible framework for health knowledge graph. In *International Conference on Smart Health*, pages 27–38. Springer.
- Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. (2020). Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th international conference on web search and data mining*, pages 672–680.

<sup>9</sup>The opinions expressed in this work do not necessarily reflect those of the funding agencies.