Generating Knowledge Graphs from Unstructured Texts: Experiences in the E-commerce Field for Question Answering

Diogo Teles Sant'Anna[1], Rodrigo Oliveira Caus[1], Lucas dos Santos Ramos[2], Victor Hochgreb[2], and Julio Cesar dos Reis[1]

[1] Institute of Computing, University of Campinas, Campinas - SP, Brazil
diogoteles08@gmail.com, rodrigo.caus@students.ic.unicamp.br, jreis@ic.unicamp.br
[2] GoBots, Campinas, SP, Brazil
lrsantostw@gmail.com, victor@gobots.com.br

Abstract. Nowadays, there is a growing number of sales occurring over the Web in e-commerce stores. Customers often have questions about a product before they buy it. By answering them instantly, online stores can improve user experience, customer's satisfaction and sales conversion rate. Effective and automated customer service via computer systems requires the handling of large amounts of unstructured information like product specifications. In this paper, we define and evaluate a technique to generate knowledge graphs (KGs) by extracting relevant product information from unstructured natural language questions and answers. The knowledge encoded in the KG is used to answer new clients' questions via SPARQL requests. Our solution is evaluated in a real world, using data from online stores in the GoBots, a leading e-commerce chatbot business in Latin America. Obtained results show the benefits of exploring KGs for responding a higher spectrum of questions in real-world settings.

Keywords: natural language processing; knowledge graphs; e-commerce; automatic question answering

1 Introduction

E-commerce stores have a keen interest in automatically understanding questions asked about their products. This can help in generating immediate and accurate answers for their customers to quickly and efficiently serve them, and also lower the cost of a large team to answer questions manually.

The answer to e-commerce questions must be highly accurate, because providing the wrong information to the customer can have unintended consequences. For example, if the customer asks if the tire works in their car and the system

mistakenly answers yes, the customer will buy the product, but will have to return it later. Moreover, if the system erroneously replies that it does not work, the customer will stop buying a product they might have purchased. To answer this question automatically, the system requires specific knowledge of whether the product is or not compatible with the car. Usually, there is a great volume of questions concerning compatibility of products in e-commerce sales.

Currently, GoBots[3] performs automatic processing of natural language (NL) questions through techniques that need to fulfill several requirements including: 1) understanding sentence intent and entities; 2) constructing vocabularies to handle synonymous terms; and 3) generating structured rules to address similar cases. However, several questions cannot be automatically answered due to lack of specific knowledge about products, specially questions regarding compatibility. Our purpose is to create a solution based on knowledge graphs to answer those type of questions and therefore complement the existent system.

In this paper, we investigate a solution to extract and structure knowledge about compatibility of products. We assume that the knowledge needed to answer untreated questions is found in pairs of questions and answers already answered by human attendants. In this sense, our solution explores as source of knowledge existing pairs of customer's questions and attendant's answers. Both components of the Q&A (question and answer) pair are in natural language and present unstructured information. Our solution extracts knowledge and structure it into a knowledge graph (KG) based on a defined domain ontology. The construction of KG adds great value because it structures information and enables further reuse and query over it.

Our approach explores the detection of entities and intentions from input questions and answers to create Resource Description Framework (RDF) triples. Our solution generates a triple store, which is used in the GoBots computational environment to help automatically answering new clients' questions. The existing GoBots solution searches for specific product information on the generated KG via API connections.

The evaluation assessed the quality of the generation of the KG for a specific domain of products and the usability of the structured knowledge to answer new questions. We use real-world data from e-commerce businesses and applied the solution to answer real-time questions to understand the benefits of the proposal. Obtained results with quality measurements indicate the promising effectiveness of our methods. Our experimental results were focused in the automotive domain, but our solution is applicable and extensible for other domains.

The remainder of this article is organized as follows: Section 2 presents the related work. Section 3 reports on our approach to generate a KG from pairs of questions and answers in NL texts and its use in the deployed KG services for obtaining answers from the KG. Section 4 shows the experimental evaluation with the results obtained. Section 5 discusses the findings whereas Section 6 draws conclusions and future work.

[3] Official website: https://gobots.ai

2 Related Work

Automated answering solutions for e-commerce is an area with a lot of research due to the high relevance of this type of solution to the market. Shiqian *et al.* [2] proposed a new framework for automatic response generation considering product-related questions through user reviews. In this framework, called RAGE, the relevant revisions made about a product are extracted based on machine learning techniques. The information is incorporated to guide response generation. Results obtained from the solution were applied to real data from online stores. Although this work proposed the generation of answers automatically, they did not investigate the structuring of knowledge about the products.

Research on the construction and use of KGs has intensified. Hoffner *et al.* [6] conducted an extensive investigation into the advances and challenges of using KGs for creation of Q&A systems. KG queries have been studied to answer questions considering structured facts expressed in the knowledge base. They complement existing work with 72 publications about 62 systems developed from 2010 to 2015. Then they identified challenges faced by those approaches and collected solutions for them from the 72 publications. Finally, they provided recommendations on how to develop future Semantic Question Answering systems.

Kertkeidkachorn and Ichise [7] introduced a framework called T2KG to create a KG automatically from natural language texts. In the T2KG, entities from natural language context are mapped to the corresponding uniform resource identifier (URI) in the KG, which are usually the subject or object of triples. In a new approach, a rule-based and a similarity-based technique are combined for mapping the predicate of a triple generated from text to its corresponding predicate in an existing KG. The experimental results demonstrated that T2KG can successfully generate a KG and populate an existing one with new knowledge from text. However, the framework performs poorly when mapping predicates containing many composite words.

Along these lines, Hao *et al.* [4] explored neural network-based classifiers to represent questions and their candidate answers retrieved from knowledge bases. In their work, difficulties of using current query languages, such as SPARQL, are presented. They argue that users need not only to be familiar with the particular language grammars, but also to be aware of the architectures of the knowledge base they are querying. By contrast, they present a question answering system relying on knowledge, which takes NL as query language, being a more user-friendly solution.

Our literature analysis indicates that there are improvements and contributions on both the fields of automatic question answering, and the usability and construction of KGs. Our work contributes with a technique to generate KGs from unstructured NL sentences organized in Q&A pairs on a e-commerce context. The method is able to extract specific information about products of a given domain, and then uses it to answer new questions with a high precision.

3 Developed approach and software tool

3.1 System overview

Figure 1 presents the KG service deployed to update and query a RDF triple store for question answering. GoBots Service (*a* in Figure 1) stands for the system responsible to manage the automatic question answering for e-commerce stores.

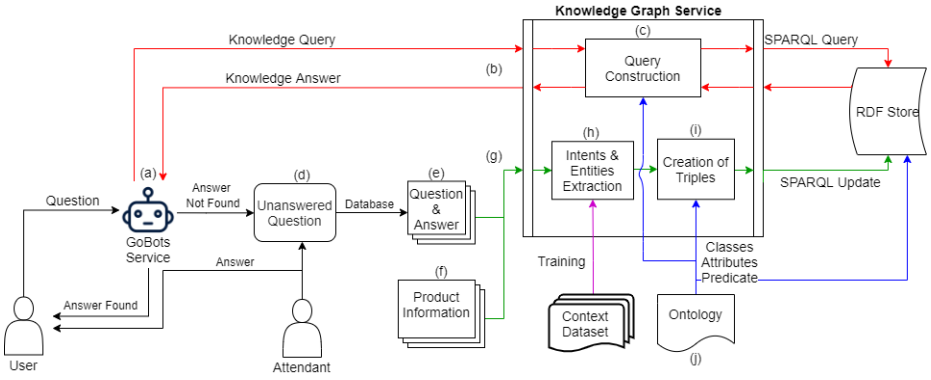

Fig. 1: Knowledge Graph Service Overview. (b) illustrates the knowledge query process to generate automatic responses on GoBots service (a). (g) illustrates the knowledge generation process [build time], regardless of (b). Its source of information is a list of questions and their respective answers from store human attendants (e). (h) is our Natural Language Understanding (NLU) process for extracting entities and intents from pairs of questions and answers. (j) is the ontology, which defines the classes and predicates to store knowledge in the RDF Store.

Formally, a KG, $\mathcal{G} = (V, E)$, is a labeled directed graph with nodes representing entities such as "Barack Obama" and edges representing relations between entities, *e.g.*, "Barack Obama" "isPresidentOf" "United States" [8]. KGs often assume the form of RDF triples in a way that $\mathcal{G} = \{t_1, t_2, ..., t_n\}$. A triple is defined as $t = (s, p, o)$ where "s", "p" and "o" are called respectively *subject*, *predicate* and *object* of the triple. The *predicate* connects the *subject* to the *object*. In the example, "Barack Obama" is the *subject*, "isPresidentOf" is the *predicate* and "United States" is the *object* of the triple t = (Barack Obama, isPresidentOf, United States). The meaning of resources in *subject*, *predicate* and *object* is encoded in a predefined ontology (cf. Subsection 3.2).

The GoBots Service is capable of querying the KG Service to automatically answer questions (*b* in Figure 1) that rely on specific knowledge stored in the RDF Store. For this purpose, the KG Service is responsible to formulate SPARQL Queries [5] (*c* in Figure 1) to retrieve knowledge from the RDF Store.

The found knowledge is sent back to the GoBots Service to formulate the proper answer for the e-commerce question.

The query requested by the GoBots Service consists of a structured representation of the components of the question asked by a customer. The key knowledge encoded in our KG refers to compatibility knowledge between products and consumer item. Then, the request contains all needed information to identify the product and a consumer item (cf. Subsection 3.2). The following information is sent to the KG service:

- the ID of the product;
- set of attributes that characterizes the consumer item;
- the intent of the question.

The SPARQL Query formulation is managed in the Query Construction component (c in Figure 1). It encodes entry data in the structure of *subjects, predicates* and *objects*, based on the ontology created for supporting question answering in the e-commerce domain (j in Figure 1) (cf. Subsection 3.2). The component manages the SPARQL result and the KG Service returns as follows:

- An indicative signalizing whether was found the knowledge of a compatibility between the given product and the consumer item as input.
- Which kind of compatibility was found relating the product and the consumer item (cf. Subsection 3.2).
- The complete attendant's answer given to the question which originated the knowledge retrieved.

In an independent and asynchronous process, the KG is updated based on a set of questions and answers handled by human attendants. A question may not be automatically answered by any of the GoBots services (d in Figure 1), so the answer manually provided by a human attendant for such question, along with the product information (e and f in Figure 1), are stored in a database for future processing.

In the process of triple creation (g in Figure 1) (cf. Subsection 3.3), the KG Service is responsible to process the stored set of questions answered manually by human attendants. This is a key source of knowledge to update the KG and answer similar new incoming questions. This knowledge extraction process relies on the extraction of intentions and entities (h in Figure 1) from each Q&A pair to structure the knowledge via RDF triples (i in Figure 1). SPARQL Updates are used to insert RDF triples into the KG, implemented using a RDF Store.

3.2 Ontology for products compatibility representation

At this stage, we present the defined ontology used to explicitly handle meaning in our KG. The ontology was created based on the e-commerce domain motivated by compatibility issues between products.

An ontology specifies a conceptualization of a domain in terms of attributes, relationships and concepts[3], the latter referred in this work as classes. Figure 2

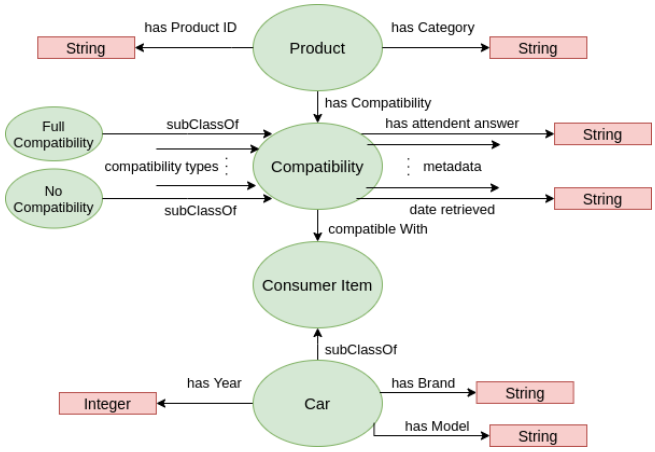

Fig. 2: Defined ontology for product compatibility representation.

presents an overview of the key classes and relationship in our ontology defined to address product compatibility issues.

The *Product* class represents a product on e-commerce. The *ID* is an identifier of the product. The ontology has different attributes to represent different kinds of *IDs*, such as *EAN*, which universally identifies products. The universal factor of the ID enables the knowledge to be reused for products of different e-commerces.

ConsumerItem is an abstract class with the purpose to identify items (owned by costumers) that may have some integration with products of the e-commerce. A subclass of *ConsumerItem* must have proper attributes that identify them. More specifically, a subclass must present a subset of attributes that uniquely identifies an Item. The importance of this subset is detailed in subsection 3.3 (concerns on minimum entities required). As an example, we present the *ConsumerItem* subclass named *Car*, representing the automotive domain, which has the attributes model, brand and year. The subset of attributes that uniquely identifies a car is model and year.

The concept of compatibility is stored through relations involving the class *Compatibility*. The usage of the compatibility as a class makes easier to store metadata about the relation between the product and the *ConsumerItem*. For example, the date the knowledge was retrieved, the e-commerce store selling the product, the question it was retrieved from and the complete answer given by the attendant.

Besides considering metadata from the relation between the product and the *ConsumerItem*, our ontology also differentiates the compatibilities between themselves throught the concept we call *compatibility type*. The compatibility type indicates whether the compatibility is affirmative, negative or conditional for example, which are represented on the ontology as *FullCompatibility*, *NoCompatibility* and *ConditionalCompatibility*. More specifically, those types are disjoint subclasses of the class *Compatibility*, which actually is used as an ab-

stract class. So every object of *Compatibility* must be object of some of the compatibility types.

3.3 Automatic KG Generation

Formally, let P be a product of an e-commerce store; id a unique identifier of P. Let $S = \{(id_1, q_1, a_1), ..., (id_n, q_n, a_n)\}$ be a set of question and answer pairs about a product, in which q_i is the question, and a_i the answer number i about the product with id_i. Also, let $\mathcal{O}$ be an ontology about the domain in which product P is inserted. We aim to automatically generate a KG $\mathcal{G}$, represented by a set of (s, p, o) triples, which expresses knowledge about the product P according to the ontology $\mathcal{O}$.

For example, let P be an armrest with a unique identifier $id = 108093$. Consider a set of question and answer pairs about P, such that, $S = \{$ (108093,"Good Morning. Does this armrest fit in Ford Fiesta Sedan year 2012 Rocam model?", "The advertised product is compatible")$\}$. Figure 3 presents this example instantiating the ontology (cf. subsection 3.2) to generate RDF triples. In this example, 17 triples were inserted in the KG (12 of them are shown in Figure 3).

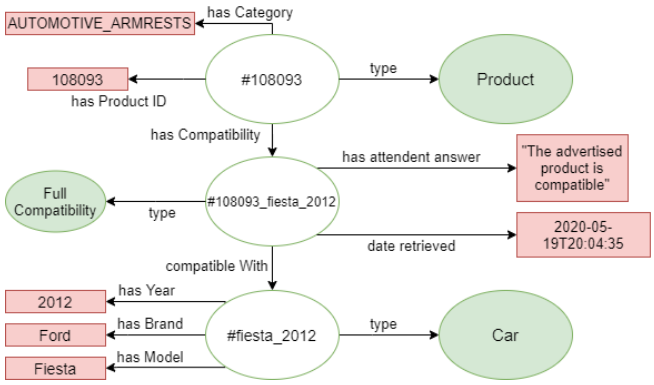

Fig. 3: Triples inserted for a question about an automotive armrest: *"Good Morning. Does this armrest fit in Ford Fiesta Sedan year 2012 Rocam model?"*.

Our solution for KG generation recognizes entities and intents (h and i in Figure 1) in sentences from questions and answers (e in Figure 1). This recognition is based on a machine learning model trained with examples of numerous entities and intents from existing sentences in the GoBots data environment.

On the NLU context, an intent represents the purpose of a user's input, which is always a sentence in natural language. Intents are a given name, often a verb or a noun, that best describes the user's intention. For example, the sentence "Is this product compatible with the car Ford Fusion 2019?", the intent "Compatibility" is identified. Our solution explores a classifier trained with several manually assigned examples of sentences with its corresponding intent.

An entity represents a term or expression with a known meaning relevant for the comprehension of the sentence. Entities have names and values. The values are the words themselves, and names represent the meaning of the word. For example, the sentence "Is this product compatible with the car Ford Fusion 2019?" encompasses the entities *brand*, *model* and *year* to identify the car. These attributes receive the values "Ford", "Fusion" and "2019", respectively. Our solution explores an entity extractor trained with names and values of entities relevant on the domain of interest.

Our solution defines intents and entities for extraction from text based on the information used to build our KG. More specifically, our technique explores question intents (from the question answer pair) to identify whether the question refers to compatibility, so an intent "Compatibility" is used. In the compatibility question, the client usually exposes the consumer item, with which the product might or not have compatibility. The question entities aim at identifying the attributes of the consumer item. In the running example, assuming a car as a consumer item, the entities "brand", "model" and "year" are used to identify attributes of the car.

For the attendant answer from the question answer pair, the technique explores the intent to evaluate the type of compatibility between the product and the consumer item. In particular, the intents used are *FullCompatibility*, *NoCompatibility* and *ConditionalCompatibility*, modeled in the ontology. Let us call this set of intents I_{valid}.

The KG generation relies on the following functions:

1. **extractEntity(F)** that for a given sentence F, returns a set $\mathcal{E} = \{(e_1, v_1, c_1), (e_2, v_2, c_2), ..., (e_p, v_p, c_p)\}$ where p is the number of entities found in the sentence; e_i is the name of the *i-th* entity found in the sentence (e.g. "year"); v_i is the value of the *i-th* entity found (e.g. "2010"); c_i is the confidence about the correctness of the *i-th* extraction [ranging from 0 to 1].
2. **extractIntent(F)** that given a sentence F, returns a pair (I, c), in which I is the intent of the sentence F, and c is the confidence of the extraction.

The solution requires the definition of a set E_{min} as the *minimal entities set* for a given domain. It is defined as $E_{min} = \{e_1, e_2, ..., e_n\}$ where each e_i element refers to an entity. These entities are the minimal ones required in a sentence (as an expected pattern). The generation of triples relies on it by assuming that such sentence carries meaningful knowledge. For our solution, let $E_{min} = \{model, year\}$ be the set of minimal entities. In this sense, if the model and year of the car are not provided, it is not possible to identify the car properly and therefore it is not possible to generate knowledge about it.

At this point, we define a minimum acceptable confidence threshold. The extraction of entities or intents requires adequate confidence level to avoid errors. Therefore, consider $minConfidence$ the value of the threshold confidence. Results with a confidence value below such value are ignored. For our solution, let $minConfidence = 0.8$.

Algorithm 1 automatically generates the KG $\mathcal{G}$ having as input a set of questions and answers made about products as $S = \{(id_1, q_1, a_1), ..., (id_n, q_n, a_n)\}$.

Triples are generated whether intent represents valid relation with a certain confidence, and entities match the expected elements from the E_{min} set with a certain confidence.

Algorithm 1 Knowledge Graph Generation

Require: $\mathcal{O}, E_{min}, I_{valid}, S, minConfidence$
for all $input \in S$ **do**
  $\mathcal{E} \leftarrow extractEntity(input.question)$
  $\mathcal{I}_{que} \leftarrow extractIntent(input.question)$
  $\mathcal{I}_{ans} \leftarrow extractIntent(input.answer)$
  if $E_{min} \subseteq \mathcal{E}.names$ **and** $\mathcal{I}_{que} = Compatibility$ **and** $\mathcal{I}_{ans} \in I_{valid}$ **and** $\mathcal{I}_{que}.confidence \geq minConfidence$ **and** $\mathcal{I}_{ans}.confidence \geq minConfidence$ **and each** $\mathcal{E}.confidence \geq minConfidence$ **then**
    $c \leftarrow instance(\mathcal{O}.getClassWithEntities(\mathcal{E}))$
    for all $e \in \mathcal{E}$ **do**
      $\mathcal{G}.newTriple(c, e.name, e.value)$
    end for
    $\mathcal{P} \leftarrow instance(\mathcal{O}.getClass(product))$
    $\mathcal{C} \leftarrow instance(\mathcal{O}.getClass(compatibility))$
    $\mathcal{G}.newTriple(C, a, \mathcal{I}_{ans})$
    $\mathcal{G}.newTriple(P, hasCompatibility, C)$
    $\mathcal{G}.newTriple(C, compatibleWith, c)$
  end if
end for
return $\mathcal{G}$

We present a full example to illustrate the KG generation procedure. Assuming the following example as input instances, S = [[ID01, Does this tire fits the palio 2014?, "Yes the product fits your car"], [ID02, "I have a ford ka 2013 sedan can I buy the product?", "Unfortunately this product doesn't fit in your car"]]. The execution of Algorithm 1 works as follows:

1. The algorithm takes the first input of S, that is, input = [ID01, "Does this tire fits the palio 2014?", "Yes the product fits your car"];
2. $\mathcal{E}$ receives the entities found by the entities extractor in the sentence input.question. In this case, input.question = "Does this tire fits the palio 2014?". For this sentence, $\mathcal{E}$ = (model,"palio", 0.85), (year,"2014", 0.92);
3. $\mathcal{I}_{que}$ receives the intent found by the extractor of intents from the sentence input.question. In this case, input.question = "Does this tire fits the palio 2014" obtains $\mathcal{I}_{que}$ = (*Compatibility*, 0.96);
4. $\mathcal{I}_{ans}$ receives the intent found by the extractor of intents from the sentence input.answer. In this case, input.answer = "Yes the product fits your car" obtains $\mathcal{I}_{ans}$ = (*FullCompatibility*, 0.94);
5. At this stage, if $E_{min} \subseteq \mathcal{E}.names$ where both are {model, year}; $\mathcal{I}_{que}$ = *Compatibility*; $\mathcal{I}_{ans} \in I_{valid}$ where I_{valid} = { *FullCompatibility*, *NoCompatibility*, *ConditionalCompatibility* }; $\mathcal{I}_{que}.confidence = 0.96 \geq minConfi-$

$dence$; $\mathcal{I}_{ans}.confidence = 0.94 \geq minConfidence$; and each $\mathcal{E}.confidence \geq minConfidence$, that is, $0.85 \geq minConfidence$ and $0.92 \geq minConfidence$, where $minConfidence = 0.8$;

6. Among the classes in our Ontology (Figure 2), it finds the class that has attributes with the same names as the entities in $\mathcal{E}.names$. In this case, the class found is *Car*;
7. The solution instantiates a car and uses an unique identifier. Instances of the same car (*i.e.*, same "model" and "year" pair) would be assigned to the same identifier. For example, let a instance have the identifier "car1";
8. Afterwards, the solution adds the attributes of that instance in $\mathcal{G}$ creating triples. In our example, the triples (car1, model, "palio") and (car1, year, 2010) were created;
9. Algorithm also instantiates the class product and the class compatibility, specifying their identifier and attributes according to the modeling in the ontology. For example, let the instances have respectively the identifiers "product1" and "compatibility1-1", the latter representing a compatibility between "car1" and "product1";
10. The final stage creates the basic triples used to represent the compatibility in the KG and adds it to $\mathcal{G}$. It obtains: (compatibility1-1, type, FullCompatibility), (product1, hasCompatibility, compatibility1-1) and (compatibility1-1, compatibleWith, car1);
11. Finally, it results in the following triples from our example: $\mathcal{G} =$ (car1, model, "palio"), (car1, year, 2014), (product1, hasCompatibility, compatibility1-1), (compatibility1-1, type, FullCompatibility), (compatibility1-1, compatibleWith, car1), (car2, brand, "ford"), (car2, model, "ka"), (product2, hasCompatibility, compatibility2-2), (compatibility2-2, type, NoCompatibility), (compatibility2-2, compatibleWith, car2);

4 Experimental Evaluation

4.1 Quality of the generated KG evaluated with question and answer in the automotive e-commerce domain

This analysis evaluates the processing of pairs of question and answers for the generation of triples to populate the KG. The questions processed were about compatibility between automotive products and customers' cars, and they were all manually answered by store attendants. The triples are inserted according to Algorithm 1 and structured based on the ontology defined in Figure 2. Our objective is to assess the capability of Algorithm 1 to generate KGs and the quality of such result.

In the experiments, we used the RASA NLU [1] to extract intents and entities from a given NL sentence through models previously trained. This framework uses a word embedding model based on *StarSpace* [9] to classify intents and conditional random fields model to extract entities. For training *RASA*, data from automotive *e-commerce* stores were collected from real-world operations in the GoBots software environment, as follows:

- List of 1,147 questions manually annotated with their respective intents for training purpose. It was divided into 28 intents, where 407 questions were classified as *compatibility* intent. The other intents were not considered to the KG generation;
- Sets of 2,054 car models, 132 car brands and 106 car years and some of their respective synonyms to be extracted from customer's question as entities.
- List of 274 real-world human attendant answers manually annotated with the respective intents. It was divided into 13 intents, where 74 were classified as *FullCompatibility* intent, 64 as *NoCompatibility* intent, and 43 as *ConditionalCompatibility* intent. Other intents were used to classify answers, but only *Full Compatibility* was used to generate knowledge at this stage of the research.

The solution was evaluated over a filtered range of questions to avoid processing questions that certainly cannot contribute with knowledge in our current solution. The complete input set S presented 25,383 pairs of question and answers. Our evaluation used all input questions that were received by one specific automotive e-commerce store between January/2020 and April/2020 that match with the following conditions:

1. The question was evaluated with the intent *Compatibility* with a confidence higher than 0.8.
2. The question was answered by an attendant from the e-commerce store.

The Algorithm 1 populated the KG with the following configuration:

- The complete input set of question and answer pairs S.
- The question intent $\mathcal{I}_{que} = \textit{Compatibility}$.
- The minimal question entities set $E_{min} = \{\text{model, year}\}$.
- The answer valid intent set $I_{valid} = \{\textit{FullCompatibility}\}$.
- Confidence threshold $minConfidence = 0.8$[4].

Considering the whole data set S as input, 1,744 Q&A pairs generated knowledge encoded in the KG[5]. In total, 20,289 new triples were added, composed by 1,534 compatibilities between products and car.

In order to understand the quality of KG generation over the whole data set S, we applied a manual evaluation on 600 Q&A pairs randomly selected from S. Three researchers (co-authors in this paper) participated on the evaluation, evaluating 200 pairs each. We inspected Algorithm 1 execution log to evaluate the correctness of the KG generation. For this, we determined two measures:

- among the Q&A pairs that generated triples, which ones generated correct triples;

[4] This minimal value of confidence was defined based on internal assessments of the solution that presented better results.

[5] A sample of the produced KG can be accessed at https://rodrigocaus.github.io/ecommerce-kgqa.

– among the Q&A pairs that did not generate triples, which ones actually had enough information to generate triples, and so should have been used to do so.

In order to obtain those measures, we made a manual evaluation inferring which Q&A pairs ideally should generate knowledge about *FullCompatibility* and which knowledge should be generated. This judgment was made without considering the NLU results. We considered the question and answer texts, judged if the question describes a *Car* with a valid model and year, and if the attendants' answer implies that the product is compatible. Let a *relevant* Q&A pair be that in which it is possible to determine both valid *Car* and *FullCompatibility* intent.

This evaluation enables to determine two metrics: Precision and Miss Rate. Precision is defined as the relation between question and answer pairs that generated correct triples by all those pairs that generated triples (cf. Formula 1). This metric signalizes the reliability of our solution.

$$Precision = \frac{\#Q\&AGeneratedCorrectTriples}{\#Q\&AGeneratedTriples} \tag{1}$$

Miss Rate is the relation between *relevant* Q&A pairs that did not generate any triples, and all *relevant* Q&A pairs (cf. Formula 2). This metric represents the amount of knowledge that should be added to the KG, but it was not.

$$MissRate = \frac{\#RelevantQ\&ADidNotGenerateTriples}{\#RelevantQ\&A} \tag{2}$$

Table 1 shows the evaluation results, from which the defined metrics were computed. The number of pairs that generated correct triples was 34 and of *relevant* Q&A pairs was 54. *Precision* was 0.971; and *Miss Rate* was 0.352. We understand that a high *Precision* and a low *Miss Rate* indicate good quality of the generated KG, although a lower *Miss Rate* is desirable.

Table 1: Results of manual evaluation of sampled Q&A pairs on KG Generation.

	Q&A pair generated triples?			
	Yes		No	
	Are the generated triples correct?		Is there enough information to generate triples?	
	Yes	No	Yes	No
#Q&A pairs	34	1	19	546

In order to evaluate the reasons that led to the generation of incorrect knowledge and missing of relevant knowledge added to KG, we classified errors on the NLP in our aforementioned manual evaluation. Figure 4 shows the most frequent *RASA* classification errors in the evaluated question and answer pairs. Note that an *incorrect pair* may contain more than one classification error. A wrong question entity (model or year) classification led to the generation of incorrect triples;

failure to correctly detect question entities and answer intents led Q&A pairs to not generate triples.

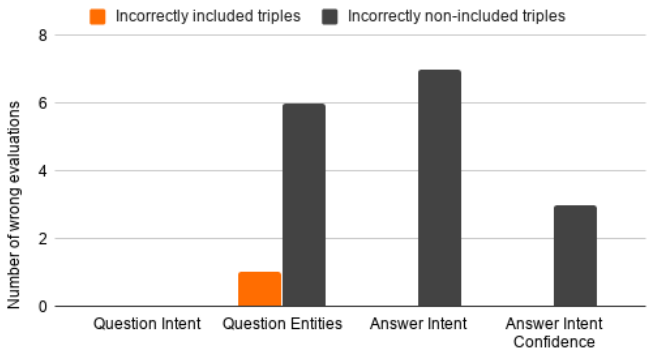

Fig. 4: Frequency of *RASA NLU* classification errors. *Incorrectly included triples* stands for *RASA* classification errors that led to the generation of incorrect knowledge. *Incorrectly non-included triples* stands for *RASA* classification errors that led to missing of relevant knowledge added to KG.

Table 1 presents 546 Q&A pairs that did not have enough information to generate triples according to our solution. The understanding of this value requires an analysis of our exigences to a Q&A pair be able to generate triples. One of our exigences, the $\mathcal{I}_{valid}$, determines that the pair can generate knowledge only if the answer intent is *FullCompatibility*. Figure 5 exposes the distribution of answer intents along the pairs of S and shows that pairs with *FullCompatibility* correspond only to 17.6% the evaluation input. This refers to the highest bound of Q&A pairs that would generate RDF triples. This shows that addressing other types of compatibility might provide ways of enhancing our KG.

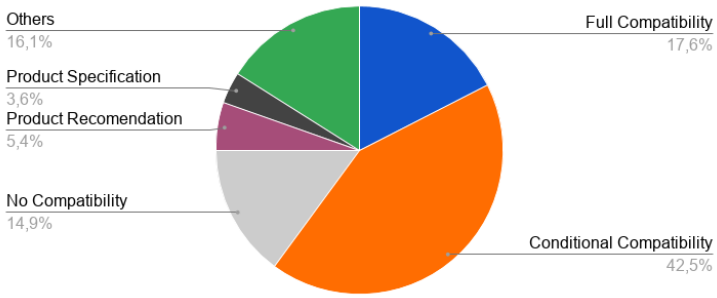

Fig. 5: Distribution of detected answer intents among the 25,383 Q&A pairs processed in S.

4.2 Evaluation of querying the KG in the automotive e-commerce domain

We deployed the KG to answer incoming questions about compatibility between automotive products and cars. The aim was to evaluate the capability of the KG Service to answer new customer questions coming from the GoBots service.

In addition to the knowledge encoded in the KG by the processing of question answer pairs, an additional set of triples was inserted in the KG from a compatibility list between cars and products provided by an automotive store working with GoBots. In total, for answering new incoming questions, the KG Service counted with a KG containing 1,923,053 triples, composed by 347,482 compatibilities between products and cars.

The evaluation was conducted on the real-world operation of the GoBots service. This receives a huge number of real-time questions to answer automatically every day. When the service is not capable to answer a given question, it queries the KG Service in an attempt to build an answer if the question fits on the criteria: the intent is *Compatibility* with confidence threshold of 0.8; and extracted entities are at least $E_{min} = \{\text{model, year}\}$.

The service was evaluated over a period of 12 days. 2,667 questions fit on the restrictions and were queried in the KG service for evaluation purpose concerning an automotive e-commerce store. It was found knowledge to successfully answer 103 of 2,667 questions posed to the KG service in the evaluated period, corresponding to 3.9%.

5 Discussion

This paper addressed the problem of extracting and structuring knowledge from questions and answers in NL on products in e-commerce. The generated KG was deployed in the GoBots systems environment and helped addressing the quality of automated answers to enhance customer services. The solution has the potential to complement the customer service that requires specialized attendants.

The solution is scalable in the sense that e-commerce stores contain a large number of Q&A pairs. The more pairs are processed, the more knowledge will be available for answering questions. Also, if universal identifier for products is applicable, knowledge from a single product can be useful and reused on different stores, given that most of the time there are intersections on items sold. Aiming to enhance the amount of knowledge available, we plan to improve the solution to process new Q&A pairs as soon as a new untreated question is answered by an human attendant.

The performance of the generation of triples is highly linked to the effectiveness of the entity and intent extraction. Figure 4 presented only one entity extraction error that led to generation of incorrect triples, which determined a satisfactory *precision* level. Most errors on NLU extraction led to missing of relevant knowledge. To decrease the *miss rate*, we should expand the training data sets and better distribute the sentences between different intentions, in order

to decrease NLU extraction errors. The production of large and balanced data sets is a major research challenge for the future development to reach further domains other than automotive.

Figure 5 presented a low number of Q&A pairs with *FullCompatibility*. To increase the number of questions answered with the retrieved knowledge from KG, we shall include *NoCompatibility* and *ConditionalCompatibility* to the valid intents to encode additional knowledge in our KG. This represents the most clear research path to follow for short-range improvements on the solution. The solution is already prepared to support such knowledge. For both *Compatibility* classes, the solution would work on an analogue way. Further improvements are necessary to encode the conditional aspect in our KG.

The proposed solution aims to work with knowledge regarding compatibility between products and generic *ConsumerItems*. Although it was evaluated using the domain of cars (automotive), our proposal is naturally and easily expansible to other domains, considering a new *ConsumerItem* CI, related to the selected domain, and the following steps:

1. Updating the ontology by adding CI and its attributes. There must be a subset of them able to uniquely identify an instance of CI.
2. Adding attributes of CI as entities on the NLU processor and training it with examples.
3. Generating knowledge using Algorithm 1 with E_{min} according to the attributes uniquely identifying CI, shown in item 1.
4. Updating GoBots Service to query KG Service when the question has *Compatibility* intent and the given E_{min} entities.

The knowledge generated by the presented solution can be valuable on different future case uses other than answering questions. For example, improving products descriptions and delivering valuable recommendations. Given the case when we find a knowledge of *NoCompatibility* for a question, our KG could be used to find another product with the same category that might have a *FullCompatibility* with the *ConsumerItem* of the question. The solution could point this other product as a recommendation.

Further improvements on the KG are also related to its potential regarding semantics. As future work, we plan to study how to take benefit of using reasoners for inconsistency detection, for instance.

6 Conclusion

Advanced solutions for customer services concerning automated question answering can benefit customer's experience and sales conversion. This work aimed to increase the effectiveness of automatic answering systems to consumer questions about products in *e-commerce* platforms. We investigated ways of generating a knowledge base encoded in a RDF triple store produced from non-structured existing data about products. The extracted knowledge has been used by the automated response system in the GoBots company to answer specific questions

without the direct assistance of human attendants. We showed the feasibility of our solution in KG construction via automatic generation of RDF triples extracted from NL messages. Experiments evaluated the effectiveness of the KG generation and asserted the high reliability of the solution applied in real-world data. We demonstrated that our solution is feasible for answering new questions based on the constructed KG. The structured knowledge was used to answer real-time questions on e-commerce store indicating its practical and direct utility. Future work involves the extension and application of the solution in a wider range of domains and with higher volume of data.

Acknowledgements

This research was supported by GoBots and the São Paulo Research Foundation (FAPESP) (Grant #2019/08609-0)[6].

References

1. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.: Rasa: Open source language understanding and dialogue management. arXiv preprint arXiv:1712.05181 (2017)
2. Chen, S., Li, C., Ji, F., Zhou, W., Chen, H.: Driven answer generation for product-related questions in e-commerce. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. pp. 411–419 (2019)
3. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (Jun 1993)
4. Hao, Y., Zhang, Y., Liu, K., He, S., Liu, Z., Wu, H., Zhao, J.: An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 221–231 (01 2017)
5. Harris, S., Seaborne, A.: Sparql 1.1 query language. https://www.w3.org/TR/sparql11-query (2013), accessed in 2020-04-04
6. Hoffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., Ngonga Ngomo, A.C.: Survey on challenges of question answering in the semantic. Semantic Web Journal **8**, 1–26 (01 2017)
7. Kertkeidkachorn, N., Ichise, R.: An automatic knowledge graph creation framework from natural language text. IEICE Transactions on Information and Systems. E101.D pp. 90–98 (01 2018). https://doi.org/10.1587/transinf.2017SWP0006
8. Su, Y., Yang, S., Sun, H., Srivatsa, M., Kase, S., Vanni, M., Yan, X.: Exploiting relevance feedback in knowledge graph search. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2015)
9. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: Starspace: Embed all the things! arXiv eprint arXiv:1709.03856 (2017)

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