2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. November 11-13, 2020

Characterization of Objects in Indoor Spaces of Human Occupation Using Knowledge Graphs

Rodrigo Francisco
Engineering Faculty
National Autonomous University of Mexico
Mexico City, Mexico
holmesrodrigo@comunidad.unam.mx

Guillermo Molero-Castillo
Engineering Faculty
National Autonomous University of Mexico
Mexico City, Mexico
gmoleroca@fi-b.unam.mx

***Abstract*—This paper proposes a knowledge graph to describe the semantic relationships of objects present in an indoor space, such as a bedroom in a house. We base our work in the representation model Object1–Predicate–Object2. In the predicate, we consider spatial relationships such as above, below, under, on top, next to, in, has, in front, and behind. To fulfill our purpose, we used the Grakn NoSQL database, which allows defining a knowledge graph schema of type entity-relation. The knowledge graph obtained needs a previously identified object in order to start looking for the correct relationships. That object is considered as input to the algorithm. The information on the semantic relationship has proved effectiveness in characterizing objects using Grakn, through which it was possible to characterize objects and their relationships based on the principles of knowledge representation.**

***Index Terms*—Knowledge graphs, Grakn, indoor spaces, objects, semantic relationships.**

I. Introduction

Within the past ten years, object recognition, a domain of artificial intelligence, has gained importance due to the significant development of hardware (CPUs and GPUs with higher performance and speed) and thanks to the constant progress of computer science (increasingly sophisticated machine learning, and deep learning algorithms). Object detection involves two main tasks: i) image classification, and ii) object localization.

First, image classification is intended to predict what kind of objects are in an image. On the other side, object localization is about locating the object with a bounding box. In general, object recognition is done with artificial neural networks, which despite being a powerful tool, they have the disadvantage of becoming slow due to the large number (millions) of node networks communicating with each other. Therefore, it is necessary to propose possible solutions to reduce processing time, such as knowledge graphs.

A knowledge graph is a set of organized information, where each vertex represents the data which can be accessed through an edge. In the same way, in [4] it is explained that a knowledge graph if an encyclopedia that machines can read. So basically it is an organized knowledge in such a way that machines can understand and extract information easily.

On the other side, [10] points out that a knowledge graph is integrated into a database that when joined, the graph is enriched and increases its significance, which allows entities and properties to be defined, which are typified and classified, then model them in your domain of knowledge. By having the domain to which it belongs, it allows another graph to connect it to another domain model.

The following example illustrates the previous idea: A system understands that a person being is an entity of soccer player type, and it is linked with soccer sport, which is played with a ball and where there are many competitions. In this way, a knowledge graph may have the following content related in a semantic way:

- Data of place, a person or a certain company.
- Images.
- Text excerpts.
- Data with secondary details.
- References to similar searches.

This paper seeks to expose the use of a knowledge graph as a tool to characterize the semantic relationships that exist between objects existing in a certain indoor space of human occupation, such as a room in a home, office, and other places of rest or work. To achieve this purpose, the model is made up of three components was taken as a basis: object1-predicate-object2. For the predicate, the relation of spatiality was considered as above, below, above, beside, in, has, in front, and others.

The document is organized as follows, Section 2 presents the background of the degrees of knowledge, their applications are discussed and related works are presented. Section 3 describes the method established as a proposed solution. Section 4 presents the results obtained, based on an example of application in a bedroom, and Section 5 summarizes some conclusions and future work.

II. Background

All recent efforts to work with object recognition successfully can be encompassed in two large model families. First, there is the use of algorithms related to convolutional neural networks (CNN), and second, there is the YOLO (You Only Look Once) approach.

Models based on convolutional neural networks, as the name implies, use neural networks with an extra layer called the convolution layer, which has the characteristic of choosing or

978-1-7281-8987-1/20/$31.00

detecting patterns and making sense of them. Such pattern detection is what makes CNN useful for image analysis. CNN works through filters, which allow detecting patterns such as circles, edges, squares, lines, among others. The famous CNN-based models are R-CNN, Fast R-CNN, and Faster R-CNN.

On the other side, YOLO, currently supported by DarkNet, is a real-time object detection system. It works by dividing the image into cells, where each cell is tasked with predicting a bounding box that involves 3 elements: X and Y coordinates, width and height dimensions, and a confidence value. With these elements it is possible to try to predict the objects found in the image, involving a single trained neural network, hence the name YOLO. In recent years, improvements have been made to this model in order to make it faster and more reliable. These improvements are called YOLO versions 2 and 3.

Regarding knowledge graphs, this field began to be known in 2012, thanks to the fact that Google decided to add a semantic improvement to its search engine, called "The Knowledge Graph". The purpose of this was to answer questions for users through analysis of what words actually mean in a query, rather than analyzing character strings. So, today it is about things and not chains [4]. From this, many companies began to create their own knowledge graphs, such as Amazon, Microsoft, Yahoo, Facebook, among others; that drive semantic searches and enable better data handling, with better data processing and smarter outputs (deliveries).

At present, there are different knowledge graphs that allow companies to create their own knowledge network, in order to improve their domain and operation. A clear example of success in the use of knowledge graphs at the Prado Museum in Spain, where the information systems of archives, libraries, and collections were exchanged for a knowledge graph. This knowledge graph is a system of representation of contents and digital resources of facts related to authors, works of art, their contents, themes, periods, and styles, as well as any object potentially related to them [9].

In this sense, bringing together two areas of knowledge, object detection, and knowledge graphs, to achieve a common objective, such as detecting an object with greater precision and using fewer computational resources is one of the great challenges posed in the current decade. Therefore, the works and publications on this topic are scarce.

Therefore, the starting point of this paper arises from the interest in the characterization and detection of objects using knowledge graphs [12], which studies computer vision for object recognition, where the objective is to identify a set of regions and thus be able to classify each section with labels. It was possible to obtain new images with unobserved contexts in previously elaborated algorithms. Managing to demonstrate that these existing algorithms could be optimized obtaining a better semantic relationship.

III. Method

In order to build the knowledge graph of the semantic relationships between the objects found in a certain indoor space of human occupation, such as a room, a deductive database [13], named Grakn, was used.

Grakn is an Open Source engine for creating knowledge graphs that allow the user to organize and model complex data networks using the Entity-Relationship scheme in its maximum expressiveness. The architecture of Grakn is basically made up of two parts, as shown in Figure 1: *Grakn* (the storage) and *Graql* (the language).

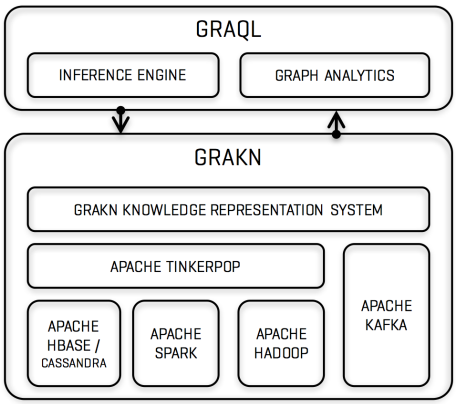

Fig. 1. Internal architecture of Grakn. Source [13].

In general, *Grakn* can be seen as a distributed, hyper-relational database that uses intuitive ontology, this is, it uses the definition of types, properties, and relationships between existing entities in a given context, to model complex data, which serves as a knowledge base for cognitive systems [14].

Thus, base on the architecture of Grakn, at the bottom we can see the system layer, which is made of Apache technologies: i) *HBase*[1] is a distributed, scalable, big data store database; ii) *Spark* for a distributed processing system used for big data workloads; and iii) *Hadoop* allows us to handle distributed processing of large data sets across clusters of computers using simple programming models. These technologies are *Tinkerpop* tools to create the structure of the knowledge graph.

On the other side, *Graql* represents the Grakn interface, where through queries knowledge can be retrieved explicitly or implicitly. In addition, Grakn was designed to be integrated with other technologies that allow it to function in a distributed manner, for example, as a complement in Natural Language Processing (NLP) and Machine Learning (ML) systems [15]. Thus, Grakn from its creation in 2016 to date is used by several companies, such as Google and Cisco; and by institutions like MIT, OpenCTI, and Cares Genetics. Table I summarizes the most important features reported by the db-engines site. A more detailed explanation can be found on the official Grakn site (https://grakn.ai).

[1]Cassandra database servers the same purposes. It is a non-relational database that is distinguished by its speed, scale, and simplicity in design.

TABLE I
GRAKN MAIN FEATURES

Feature	Description
Database model	Graph DBMS and relational DBMS
Initial release	2016
Current version	1.7.2, June 2020
Origin country	United Kingdom
License	Open Source
Implementing language	Java
Supporting operative systems	Linux, OS X and Windows
Triggers	no
Supporting languages	All languages base on JVM: Groovy, Java, JavaScript (Node.js), Python, Scala
APIS and other access methods	Console (shell), gRPC protocol, Workbase (visualisation software)
Technical docs	https://dev.grakn.ai/doc
Usually compare with	Neo4j, GraphDB y JanusGraph

A. Grakn installation

To install Grakn, Java 8 SDK is previously required, if not you can download Oracle or OpenJDK implementation. To install Grakn in Linux we can use three package managers: `dnf`, `yum` (for systems that uses RPM[2]) and `apt`. Besides, Grakn is also available for X OS operative system, through `brew` package manager. For this paper, the testing implementations were made using CentOS 8.2 and Manjaro Linux 20.0.

Another important point is that Grakn works similarly to database management systems, that is, services must be started, users must be created, workspaces must be created, among others. In Grakn the abstract models that encapsulate the entities of a particular problem are known as *Keyspace*, where the *schemes* that will represent the data of the case study are defined.

An *scheme* is an inherent part of the knowledge graph that describes what data is like and how it can be structured. It can be represented by an Entity-Relationship diagram. On the other side, Grakn has various types that made up the core system and provide the vocabulary necessary to describe any case study. Among the main types that Grakn handles, the following stand out:

- *Entities*. They provide the means to classify the objects of a given domain.
- *Relationships*. They connect the elements, known in Grakn as *things* from a particular domain. They can be objects, relationships y attributes.
- *Attributes*. They describe the entities.

In Graql we can perform data manipulation and data definition operations. Likewise, it is important to mention that the query language *Graql* has the following characteristics, which were useful for this work.

- *Declarative*. In order to make a query with Graql, it may describe what you want to recover, instead of saying how it should be obtained.
- *Intuitive*. Graql was designed to provide a high-level query language interface with clear, readable syntax.

[2]RPM is a recursive acronym which means RPM Package Manager

B. Semantic relationships schema

The semantic characterization created in this work is based on the structure: `object1--predicate--object2` proposed in [16], who points out that visual relationships capture a wide variety of interactions between pairs of objects in images. Therefore, it is recommended to extract the possible relationships that may exist, in order to reduce them and work more efficiently. Besides, the relations between each pair of objects are required to have a predicate individually, and based on this a classification is made to generate a relation. The most common types of predicates are classified by an action, space, a preposition, a comparison, or a verb as seen in Figure 2.

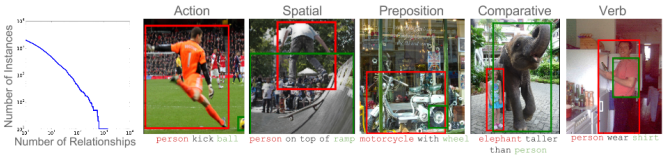

Fig. 2. Predicates categorized. Source [16].

Based on the above, in this work, the characterization of objects by spatial predicates was performed, since the objects found in a certain space of human occupation, such as a room, are static. Whereas, in this case, predicates based on verbs (actions) were not useful. In the Figures from 3 to 7 the relationships used to make the knowledge graph are shown.

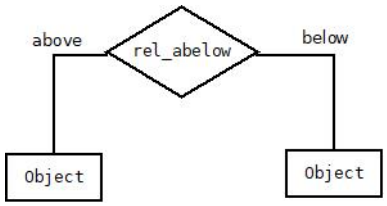

Fig. 3. Relation above-below.

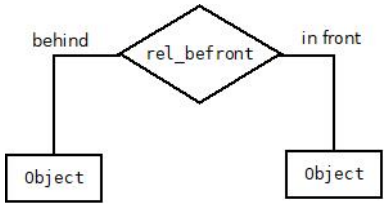

Fig. 4. Relation behind-inFront.

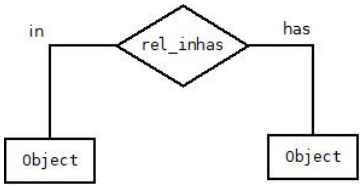

Fig. 5. Relation in-has.

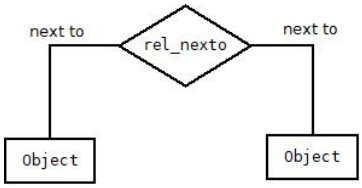

Fig. 6. Relation nextTo.

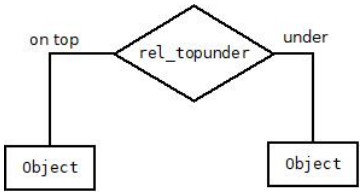

Fig. 7. Relation onTop-under.

The elements recognize as objects have two attributes which are the identifier (ID) and the name as shown in Figure 8. However, as more predicates are incorporated, it is possible to add more attributes to the object in order to better describe the relationship.

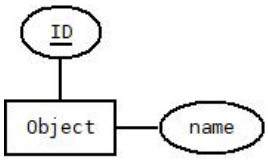

Fig. 8. Object attributes.

Before implementing the knowledge graph with Grakn, we did a draft of it, in order to visually identify the relationships that must be created through the tool. This conceptual design is seen in Figure 9.

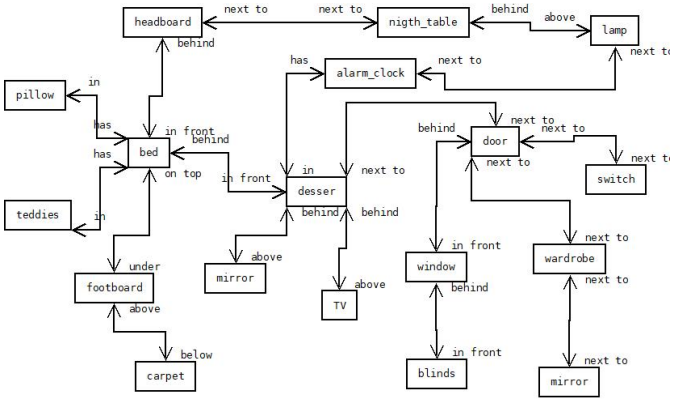

Fig. 9. Conceptual design of the knowledge graph.

C. Development

The implementation of the knowledge graph in Grakn was based on the previously defined relationships scheme, shown in the previous section. Thus, with the help of the Graql language, relations of type `above-below` were defined first and then the object, as can be seen in Listing 1 and 2, respectively.

```
define
rel_abelow sub relation,
  relates above,
  relates below;

rel_nexto sub relation,
  relates next_to,
  relates next_to;
```

Listing 1. Relationship definition using Graql.

```
object sub entity,
  plays above,
  plays below,
  plays next_to,
  plays behind,
  plays infront,
  plays ontop,
  plays under,
  plays in_,
  plays has_,
  has name;

name sub attribute,
  datatype string;
```

Listing 2. Object definition using Graql.

Later, we insert the object and its relationships data with the code seen in Listing 3, for inserting the objects and Listing 4 to insert the relationships.

```
insert $obj0 isa object,
  has name "bed";
```

Listing 3. Object insertion.

```
match
  $o isa object, has name "bed";
  $b isa object, has name "pillow";
insert $rel (in_:$o,has_:$b) isa rel_inhas;
```

Listing 4. Object relationship.

On the other side, to execute the code it was necessary to create a dedicated *keyspace*, which was named `vision_relationships`, where the schemes and relationships were made. Finally, to check that the objects have been inserted correctly, a query was made via console, as shown in Figure 10. Thus, the created objects were also counted.

```
proyecto> match $obj isa object,has name $na;get;
{$na "dresser" isa name; $obj id V4208 isa object;}
{$na "wardrobe" isa name; $obj id V4256 isa object;}
{$na "footboard" isa name; $obj id V4272 isa object;}
{$na "bed" isa name; $obj id V4320 isa object;}
{$na "pillow" isa name; $obj id V8288 isa object;}
{$na "teddies" isa name; $obj id V8304 isa object;}
{$na "alarm_clock" isa name; $obj id V8384 isa object;}
{$na "blinds" isa name; $obj id V8416 isa object;}
{$na "window" isa name; $obj id V12384 isa object;}
{$na "mirror" isa name; $obj id V12480 isa object;}
{$na "lamp" isa name; $obj id V16416 isa object;}
{$na "switch" isa name; $obj id V20672 isa object;}
{$na "door" isa name; $obj id V24768 isa object;}
{$na "TV" isa name; $obj id V28896 isa object;}
{$na "headboard" isa name; $obj id V40968320 isa object;}
{$na "nigth_table" isa name; $obj id V40968408 isa object;}
{$na "carpet" isa name; $obj id V40972416 isa object;}
proyecto> compute count in object;
17
```

Fig. 10. Query to check that there are no errors.

IV. Results

Figure 11 shows the knowledge graph generated by the query shown in listing 5. It is appreciated that the rectangles in purple contain the name of the object and those in fuchsia represent the identifier of the object. Each object can have zero or more relationships to other objects[3]. However, an object unrelated to others is of no use because it could not be reached since it has no edges, which represent ontological relationships. In addition, implicit or explicit information can be extracted through the edges (green lines).

```
match
  $obj isa object,
    has name $na,
  $r1 ($z,$y) isa rel_nexto;
  $r2 ($x,$w) isa rel_topunder;
  $r3 ($v,$u) isa rel_inhas;
  $r4 ($s,$r) isa rel_abelow;
  $r5 ($q,$p) isa rel_befront;
get;
```

Listing 5. Query to generate the knowledge graph.

[3]Strictly, they only relate to objects that are part of the query.

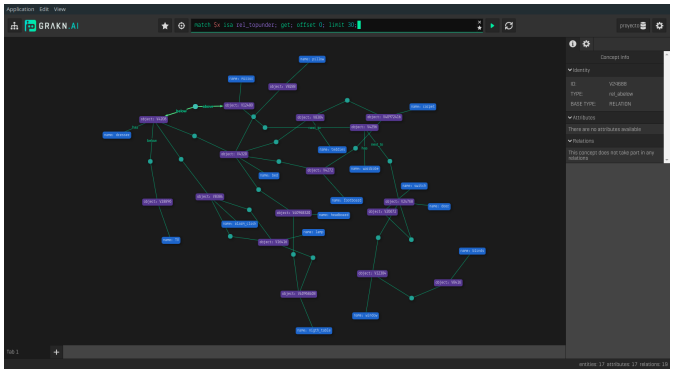

Fig. 11. Knowledge graph in Grakn.

The explicit information obtained from the edges explains why the objects were determined to be related. In most cases it is by the nature of the *query* that it is executed. On the other hand, the implicit information is related to the *background* of the relation of objects and is usually used when making inferences, better known as *automatic reasoning*. It is worth mentioning that in this work we only work with explicit relationships.

Figure 12 shows the relationship *rel-topunder* between the bed and footboard which can be expected while finding the object "footboard" under the object "bed" and the object "bed" in top of the "footboard". Also, in the same image, the *rel-abelow* relationship between the footboard and carpet can be seen similarly to the previous one.

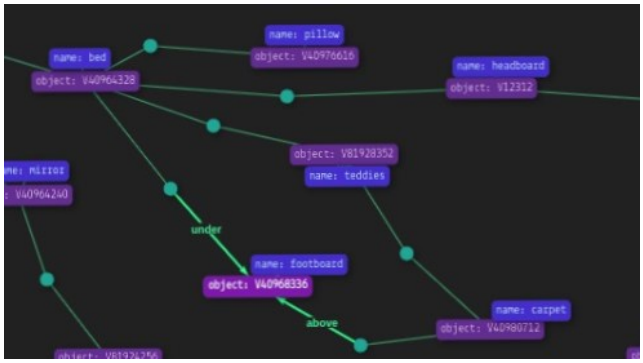

Fig. 12. Relation bed-footboard, footboard-carpet.

This work is a contribution to the categorization of the semantic context in computer vision since it establishes the tools and a baseline in spatial relationships to create maps of the environment of places of human occupation, using knowledge graphs, in contrast to the common way it is done.

Normally, to avoid processing an immense quantity of convolutions, which translates into higher execution time, [18] suggests creating object context base on object scale or spatial context, by creating big comparison lists (dataset) and contrasting them with image training sets. This is a good approach, however, the list remains static and does not get feedback from the training dataset. Which our model does naturally because that is how knowledge graphs work.