SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL AND MANAGEMENT ENGINEERING

The Curiosity Bot

HUMAN-ROBOT INTERACTION AND REASONING AGENTS

Professors:
Luca Iocchi
Fabio Patrizi

Students:
Massimo Coppotelli
Andrea Giuseppe Di Francesco
Riccardo Riglietti

Academic Year 2022/2023

Contents

The authors of this report have equally contributed to the realization of the project.

1 Introduction

The education of young children is fundamental to the progress of society, and it is proved that hands-on learning empowers their curiosity and improves the level of attention of the youngest. Since a shortage of teachers [1] still affects many developed countries, we have thought of a Curiosity Bot that placed in several contexts (e.g., in libraries, restaurants, hospitals) can interact with children, satisfying their curiosity on everyday objects. We provide a novel way to interact with robots to make learning more fun and effective [2, 3].
To develop this project, we decided to rely on the Pepper robot's architecture [4] because its physical appearance (see Figure 1) is intentionally designed to cultivate empathy with the user and foster a curiosity-driven interaction with humans. We firmly believe that this deliberate choice avoids the "uncanny valley" [5], effectively enhancing the user's curiosity during the interaction.

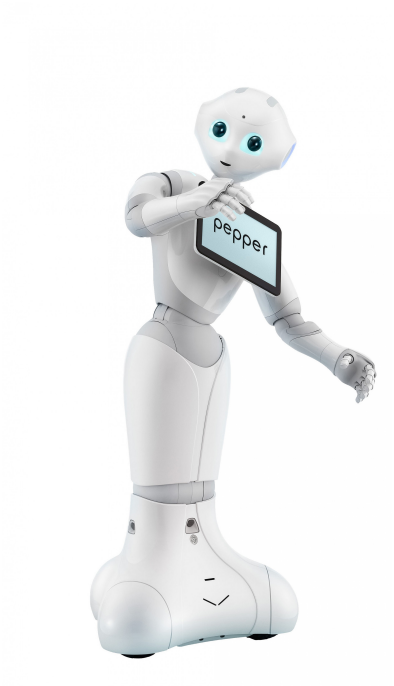

Figure 1: Appearance of the Pepper's robot.

The Curiosity Bot should be placed behind a table with a few objects on it; Figure 2 shows a top-down view of this setting.

A typical interaction with the robot can be summarized as follows:

1. Pepper asks the children to show any object;

Figure 2: Top down view of the Pepper robot, he must be put behind a small circular table with already known objects on it. The child will approach the robot from the front and show it any object, touching Pepper's head to shoot a photo and classify the object. The robot will then point to the table's object that is the most similar to it, making use of the physical props that it has access to.

2. Pepper recognizes the object and provides a description of it;

3. Pepper connects the object shown with the most similar item on the table and formulates a logical sentence to show their connection.

This basic interaction loop is animated by an ask & reply-like conversation between the robot and the child.
Our work takes advantage of *Human-Robot Interaction* (HRI) techniques to interface with the user in a natural way: rotating to face the correct object, pointing at objects, and giving the interlocutor a chance to interact, with the intent of generating a non-linear dialogue. On the other hand, *Reasoning Agents* (RA) techniques, were used to generate the sentences that relate the object shown by the user and an object on the table. This is accomplished by automatically choosing a path to connect their synsets in the WordNet [6] graph.

2 Related Works

2.1 Human-Robot Interaction

As mentioned in the introduction, introducing a robot that has the capability to entertain children in a constructive and interactive way is an idea with a great potential, especially when considering the shortage of teachers in developed countries. A didactic robot such as this one could be placed in a public library and interact with hundreds of children every day, increasing their interest in both general curiosity about the world and technology.

A success-full experiment described in the study "A Robot as a Teaching Assistant in an English Class" [7] utilized a small radio-controlled robot to improve the engagement of the children in an English class. Language classes are particularly suited to creative ways of teaching because they can contain any kind of content as long as it is in the target language. In the study a small radio-controlled robot makes dance moves and cheers the students up when they give a correct answer. The large difference between this work and ours, is that our robot is fully autonomous, and it takes on the role of the teacher rather than the role of the teaching assistant, thus being able to be used fully autonomously.

A critical aspect of successfully human-child interaction is that the robot should present itself as a peer and a friend rather than as an authority figure. As mentioned in the study "Fragile Robot: The Fragility of Robots Induces User Attachment to Robots" [8] users develop a stronger sense of sympathy and "friendship" with robots that have some flaws and/or uncertainty, either physical or in their behaviour. Because of this we introduce uncertainty and epistemic humility in the way our robot handles the image recognition results: if the confidence of the prediction is lower than 95% the robot will ask the child for confirmation if the guess of which object has been shown is right or not. In practice this threshold could well be lower, but we prefer to ask more rather than less, as an uncertain and thus "fragile" robot is more friendly.

Counter-intuitively, it is also important to avoid being too friendly in a didactic setting, as a robot that is too chatty, customized and friendly will surely be an enjoyable experience for the child, but it can sadly reduce the amount of information that the children learns, as discussed in the paper: "The Robot Who Tried Too Hard: Social Behaviour of a Robot Tutor Can Negatively Affect Child Learning" [9]. Because of this we keep the human-robot interactions strictly on topic, with minimal purely social interactions (only greeting the user at the start and at the end of the interaction).

Finally the great literature review: "Social robots for education: A review" [10] explains in detail the many lessons learned from the successes of many experiments of didactic robots around the globe. This meta-review has the really positive finding that almost all application of social robots for education aimed at increasing learning outcomes had a positive effect, thus underlining the great positive potential of this

technology. We highly suggest it as further reading for any reader interested in the subject.

2.2 Reasoning Agents

Pepper must find a way to explain the connection between the object shown by the user and the closest object by the table. This can be done with the CLIP (Contrastive Languge Image Pretraing) [11] model, which is a multimodal encoder that encodes both images and text in a shared latent space, with similar concepts mapped to similar vectors. The closest object on the table is chosen as the one with the CLIP text embedding closest to the CLIP image embedding shown as input.

This process has a high performance given the large scale pre-training of the CLIP model but is not interpretable, that is, the model cannot give an explanation for why the vectors are close in the latent space, they just are. In order to overcome this problem and give a fact checked and reliable explanation of the choice of the most similar object on the table to the user, we use the resource of WordNet [6], combined with the A* algorithm [12] to find a path from the synset of the object shown by the user and the synset of the object chosen by CLIP as the most similar one. Further details are provided in Sections 3 and 4.

We also tried using the FF [13] solver by converting WordNet into PDDL but it does not find the path in a reasonable time given the complexity of the grounding step and the size of WordNet (117000 synsets).

3 Solution

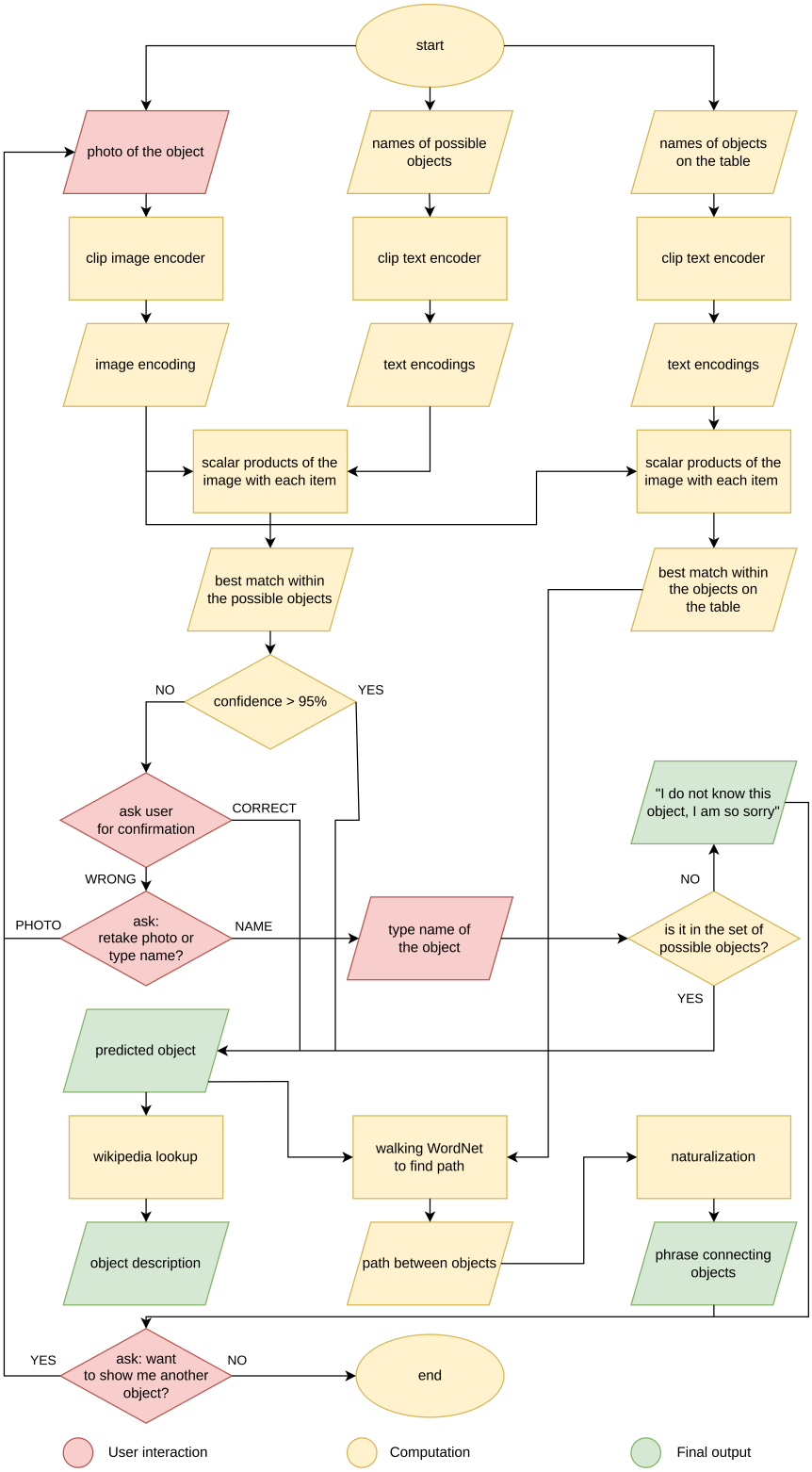

Figure 3: A full explanation of how the different modules are connected in order to support the interaction. Before taking the first photo Pepper asks the user for its age, if the user is less than 10 years old, then the "children mode" is enabled and the "object description" output is shortened (omitted from the diagram to avoid clutter).

In this Section, we provide details on the architecture of the solution, highlighting the role of the individual components, and how they communicate among each other. Our description splits the HRI and RA aspects in different sections for clarity.
Figure 3 reports a high-level overview of how the components of our system are connected to each other.

3.1 HRI

The Human-Robot-Interaction part of the project consists of integrating all of the modules shown in Figure 3 in a simple, clear, and interactive way.

The interaction starts in a standard way with our robot introducing itself and stating its purpose, than the robot asks the user (i.e. a child) to show it any kind of object.

The image recognition is handled by the CLIP (Contrastive Language-Image Pre-Training) [11] neural network that is trained on a variety of (image, text) pairs; this allows it to perform zero-shot, open-label image classification, making it possible to easily expand the system to support any new object without any kind of fine-tuning.

We take into account the importance of Fragile Robotics [8] by making sure to check the confidence of the prediction and asking confirmation to the user in case the confidence is not extremely high (we ask for confirmation if $p < 95\%$). We ask the user if he wants to retake the photo or if he wants to directly say the name of the object; if he wants to take a photo, the robot gives standard suggestions to the user in order to help him make a better photo, by suggesting to keep the object steady and to improve lightning conditions: a simple form of human-robot cooperation to achieve a shared goal.

Then, some relevant information about the object is retrieved from the Simple English Wikipedia, and the wikitext is cleaned from the formatting in order to get a clean text to present to the user, that will be spoken by Pepper.

Finally the connection between the object shown and the most similar item on the table is explained as mentioned in the next section. The HRI part of this explanation is that the types of relations between the synsets are translated from highly technical terms (Hypernym, Meronym, Holonym, Hyponym) to simple prepositions (see Table 1 for the phrases used).

3.2 Planning

3.2.1 Explanation of the connection, step by step

CLIP embeddings similarities are used to identify the object from the table that is the most similar to the one chosen by the user, then the connection between them is found by exploring the WordNet graph [6].

WordNet [6] is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. The synsets are inter-connected by conceptual-semantic and lexical relations.

3.2.2 Solution with formal planning and FF solver

We converted the WordNet graph into formal language by using the predicate ‘at‘, ‘go-type‘, and ‘connected-type‘, then using the FF-solver [13] to solve it. Section 4 illustrates how WordNet was translated into planning language.

The solver has great efficiency problems given the complexity of the grounding step, so we decided to move to a custom A* implementation for the traversal of the semantic graph.

3.2.3 Solution with A*

Given that the problem is just finding the shortest path from the initial node to the final node given a certain list of connections, the A* algorithm fits perfectly with this task. Given the high connectivity of the WordNet graph, the path between the two synsets can be found in real time in around one second or less, giving a satisfying real time interaction with the user. The heuristic function $h(\cdot)$ that has been implemented for the A* is based on GloVe embedding similarity [14]. It is formalized in Equation 1.

$$h(syn_{ref}) = 1 - cosine(GloVe(ref), GloVe(target)) \tag{1}$$

$h(syn_{ref})$ is the underestimated distance from syn_{target}, while $cosine(\cdot, \cdot)$ computes the cosine similarity of the GloVe embeddings related to ref, and $target$.

Semantically similar words have a high GloVe embedding similarity, by swapping the sign we obtain an estimate of the "cost", that is the semantic distance between words. We add one in order to keep the heuristic function positive.
Given the property of Equation 2, Equation 3 derives.

$$0 \leq cosine(GloVe(ref), GloVe(target)) \leq 1. \tag{2}$$

$$0 \leq h(syn_{ref}) \leq 1. \tag{3}$$

This heuristic is admissible and consistent as it always predicts a number less than one, and the minimum distance between nodes is one, hence the consistency, and the distance to the goal is always more than one, hence the admissibility. The choice of using GloVe embeddings is the most practical, with respect to a deep-learning-based solution such as CLIP embeddings because large hashmaps of (word, GloVe emebdding) can be downloaded freely from the internet while creating these pairs for a large transformer model for a very large number of words is relatively computationally

expensive.
Our A* implementation keeps track of the nodes and of the explored connections, in order to build a natural language phrase to show to the user.

Technical Name	**Explanation**	**Short Text**
Hypernym	A higher-level category or class	is a
Meronym	A part or component of something	contains
Holonym	Something that includes or comprises other parts	is a part of
Hyponym	A more specific instance or subclass	could be a

Table 1: Terminology definitions for the WordNet

3.3 Examples of paths between synsets found by our A* implementation

In this section we show two examples of how our method connects together some common objects, creating logical and easily understandable phrases.
For example, when we want to find the connection between "pen" and "pencil" we can walk along the following connections:

1. We connect "pen" with "writing implement" because "pen" is a "writing implement" (technically we are moving from "pen" to its hypernym "writing implement")
2. We connect "writing implement" with "pencil" because "writing implement" could be a "pencil" (technically we are moving from "writing implement" to its hyponym "pencil")

This can be seen graphically in Figure 4.

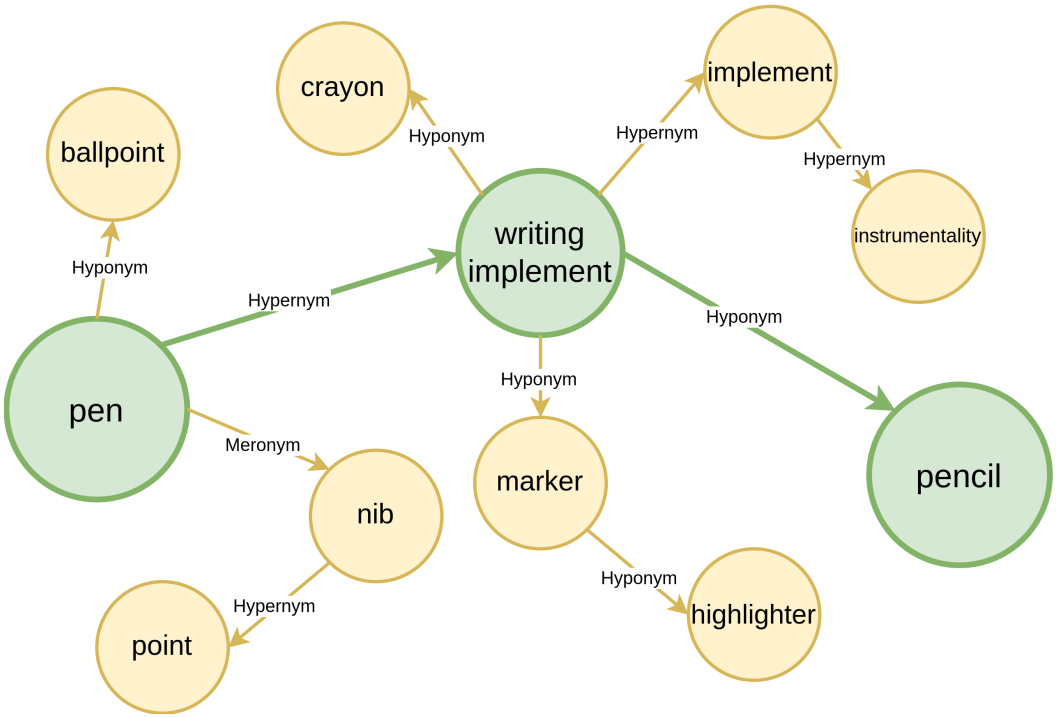

Figure 4: A graphical representation of a small subset of WordNet. It shows how "pen" and "pencil" can be connected by following the relations between synsets that are present in WordNet (relevant nodes and relation in green), together with a small number of nodes and relations that are not relavant for the purposes of this connection (in yellow).

Another more complex example is when we want to find the connection between "mouse" and "computer":

1. We connect "mouse" with "electronic device" because "mouse" is a "electronic device" (technically we are moving from "mouse" to its hypernym "electronic device")

2. We connect "electronic device" with "device" because "electronic device" is a "device" (technically we are moving from "electronic device" to its hypernym "device")

3. We connect "device" with "machine" because "device" could be a "machine" (technically we are moving from "device" to its hyponym "machine")

4. We connect "machine" with "computer" because "machine" could be a "computer" (technically we are moving from "machine" its hyponym "computer")

This can be seen graphically in Figure 5

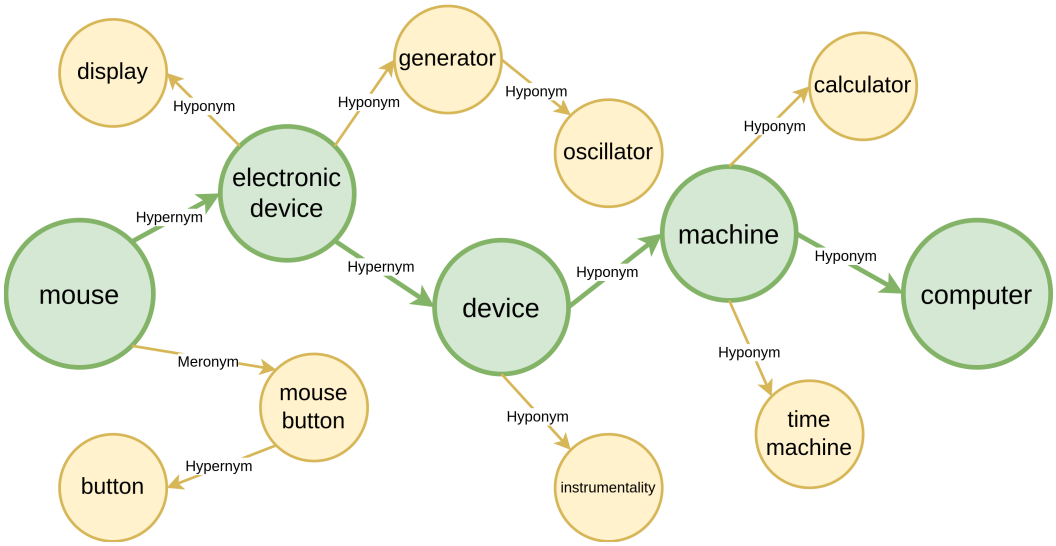

Figure 5: A graphical representation of a small subset of WordNet. It shows how "mouse" (with the meaning of "computer mouse") and "computer" can be connected by following the relations between synsets that are present in WordNet (relevant nodes and relation in green), together with a small number of nodes and relations that are not relevant for the purposes of this connection (in yellow).

4 Implementation

As mentioned in the introduction, here is a high level view of the steps of the interaction in this project, now divided in smaller parts:

1. Ask the children to show any object.
2. Recognize the object.
3. Give a description of it based on Simple Wikipedia.
4. Connect it with an item on the table and show the logical steps of the connection by using WordNet [6].

This section will now explain in detail how each part has been implemented, including the libraries and deep learning models used.

4.1 HRI

4.1.1 Ask the children to show any object.

In this part, the robot greets the user (using both voice and phisicality), presenting itself as the "Curiosity Bot", and asks for his/her age. If the user is under 10 years old, the curiosity bot enters in "children mode", providing shorter explanations of the objects that the child will show during their interaction. At this point, the robot asks the user to show it any object.
This part of the interaction is handled by sending hardcoded messages and realized by the text-to-speech function of the Pepper API, that are received by the Android Studio environment with the specialized extension.

4.1.2 Recognize the object

In order to recognize the object the CLIP model [11] is used. CLIP (Contrastive Language-Image Pre-Training) is a neural network trained on a variety of *(image, text)* pairs, that we have chosen for its high performance and zero-shot classification capabilities. The advantage of it is that all common objects can be classified without the need for fine-tuning, being able to deal also with custom classes (like "small red cube", "big blue pyramid"). This model is provided with its pretrained weights in the `transformers` library, that we installed in python3. The python3 scripts for classification are called from the python2 scripts as a way to handle the need for advanced and more recent state-of-the-art methods, due to the incompatibility between Pepper and python3 as well as the incompatibility between the `transformers` library and python2.
CLIP embeds images and text in a shared space, where the distance between embedding

is based on the semantic distance between the concepts, so the embeddings produced from the photo of a dog and from the string "dog" will be close in this latent space. We compute the dot product between the embedded image and the embedded labels that we have chosen as possible objects, in order to perform the classification task.

4.1.3 Give a description of it based on Simple Wikipedia

Once the classification of the image is carried out, it is important to provide a description of it. To accomplish this, we have been taken advantage of the open-source XML dump of Simple-English Wikipedia.
We downloaded an XML dump of the Simple English Wikipedia [15] and used the `xml` Python library in order to parse all the articles contained inside. This creates a hashmap that connects the title of each Simple English Wikipedia article to its contents, allowing us to retrieve relevant information about almost all possible objects in real time.

Before showing the result to the user a series of *Regular Expressions* (REGEX) are used in order to clean the text from markdown-like formatting typical of wikitext and obtain a plain text readily available for the Human-Pepper interaction. We underline that the whole textual Simple English Wikipedia can be downloaded without any problem as its size is only about 1GB.

4.1.4 Connect the object with the most similar one on the table and explain why they are connected

We use the scalar product of the CLIP embedding of the image shown and of the names of the objects on the table to find the more similar one. Then we find an explanation with the RA technique explained in the following section and translate the explanation into simple language using the Table 1.

4.2 RA

4.2.1 Show the logical steps of the connection by using WordNet

A WordNet synset refers to a group of data elements or words that are semantically equivalent in the context of information retrieval. In other words, a synset represents a collection of synonyms that can be used interchangeably without altering the truth value of the proposition in which they are used. A synset serves the purpose of organizing and associating synonymous terms for effective information retrieval and semantic equivalence.
Starting from the initial synset, that is the one matching the object recognized by CLIP, and the final synset, that is the one matching one of the table objects retrieved by means of the scalar product of the CLIP embedding of the photo of the object

shown by the user and the CLIP embedding of the name of the object on the table, our planning strategy had the objective of searching for the shortest possible path that connects them.
This specific problem was formulated as an informed search problem on a graph. Then, two approaches have been implemented: an approach based on the PDDL language and the FF planner and a "custom" A* implementation, that was necessary for efficiency.

4.2.2 PDDL and FF planner

The problem of finding the shortest path between two WordNet synsets is analogous to the problem of solving a maze, with the difference that two cells of a maze are connected if a physical corridor connects them, while two synsets of WordNet are connected if a conceptual relation exists between them, for example the synset "calculator.n.01" is connected to the synset "machine.n.01" with the relation of "hyponym" because a calculator is a type of machine.
In analogy with the maze example, we define the following predicates, with "at" that contains the current cell of our exploration and "connected-[type]" that means that two synsets are connected by a relation of type [type], finally we have the actions "move-[type]" that allows the exploration to move from one node to another if they are "connected-[type]". We have a different predicate and a different action for each type in order to be able to reconstruct the kind of connections that we encountered between synsets when building the path. Here is the domain PDDL file with the repetitive parts ellipsed with "...":

```
(define (domain connected-nodes-kinds-domain)
  (:requirements :strips)

  (:predicates
    (at ?loc)
    (connected-related ?from ?to)
    ...
    (connected-cause ?from ?to)


  )

  (:action move-related
    :parameters (?from ?to)
    :precondition (and (at ?from) (connected-related ?from ?to))
    :effect (and (not (at ?from)) (at ?to))
  )
   ...
   (:action move-cause
    :parameters (?from ?to)
    :precondition (and (at ?from) (connected-cause ?from ?to))
    :effect (and (not (at ?from)) (at ?to))
  )
)
```

The problem file is built automatically from WordNet, here is a small handmade example (the file containing the whole WordNet has over 117000 objects, with even more relations, so the complete problem file is huge, and not practical to use because of the complexity deriving from the grounding step):

```
(define (problem connected-nodes-kinds-problem)
  (:domain connected-nodes-kinds-domain)

  (:objects
    fish boat water vehicle animal
  )

  (:init
    (at fish)
    (connected-related-to fish water)
    (connected-related-to water boat)
    (connected-is-a fish animal)
    (connected-is-a boat vehicle)
  )

  (:goal
    (at vehicle)
  )
)
```

Given these files, we can run the FF planner (we used the library `pyperplan` as the implementation of the planner) in order to get the following path, that contains information on both the synsets that have been visited and the relations connecting them:

```
(move-related-to fish water)
(move-related-to water boat)
(move-is-a boat vehicle)
```

The FF planner is a planner that estimates the distance from the goal by creating a simpler version of the problem. This simplified version doesn't consider the "delete lists" of the actions involved; in other words, we can consider that we can gain all the positives from our actions without having to pay any costs. This is a clearly optimistic (admissible) heuristic that can be useful in the search process.
Applying the FF planner on the two PDDL files mentioned above works correctly, and it also works correctly on many other simple examples not included for brevity, but would require a very long time on the full WordNet as previously mentioned.
The grounding step is responsible for generating a ground program that does not contain any variable but has the same answer sets as the original program, and it has exponential time complexity due to the huge number of substitutions that must be made [16], as such, given the uniformity of our problem (all the relations are connections between synsets "connected-[type]", all the actions are movements from one node to another "move-[type], and we have a predicate "at" containing the current

position"), we do not need to perform this very complex step and we can instead use the A* algorithm.

4.2.3 A*

Given the analogy of WordNet with a physical maze mentioned above, it is clear that A* is an appropriate algorithm to solve this problem, with the difference that in the case of finding the path in WordNet it is harder to use a heuristic, while the distance of two cells in a physical maze can clearly be optimistically approximated by the euclidean distance of their position in space, it is very hard to approximate in a simple way the semantic distance of two synsets in units of WordNet relations needed to connect them. Nonetheless, we attempted to provide A* with a heuristics that relies on the GloVe's embeddings similarities. We have seen its details in Section 3.
This ad-hoc implementation of A* avoids the need for a grounding step as in the PDDL solution, and as such it is much faster, solving almost all paths in less than a second (that is close enough to real time for the purposes of human robot interaction).

5 Results

In order to handle the uncertainty of the real world and the incompleteness of our knowledge bases (both Simple Wikipedia and WordNet), we have many different paths that the interaction can take based on how recognizable the image is and if we find relevant information or not in our offline database of simple English Wikipedia, as shown in the logic flow diagram in (Figure 3).
In the following sections, we will describe some examples of interactions, showcasing how the robot can adapt to the complexities of the real world, responding gracefully to both uncertainty in recognizing the object and to not finding relevant information about the object in our offline dump of Simple Wikipedia.

5.1 Interaction with clear image and all information found

This is the best case of the human-robot interaction, where the image is clear and the relevant information is found in Simple English Wikipedia. In this case, if the confidence of the image recognition is over 95% the robot will simply state that it has recognized the object shown without any doubt and it will give the relevant information from Wikipedia to the child.

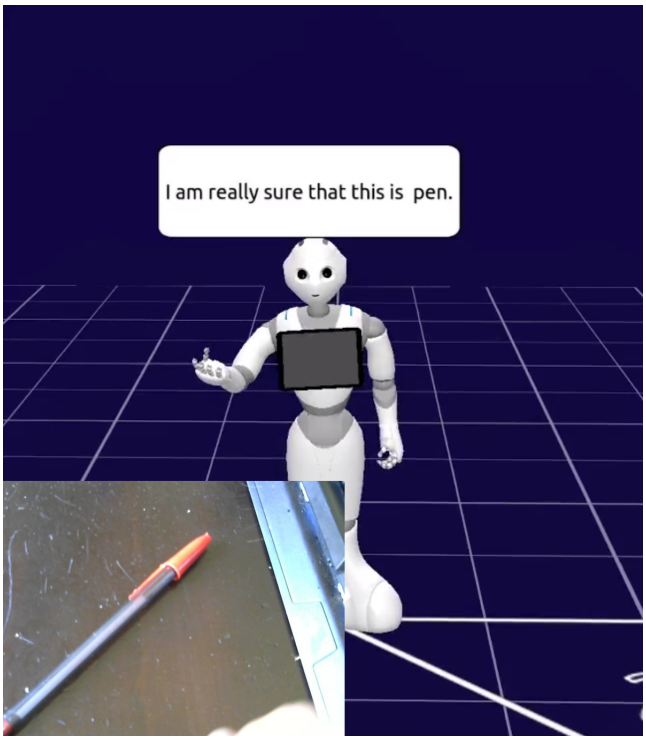

Figure 6: Pepper in this case is very sure of its prediction ($> 95\%$) so he does not ask for confirmation as it would only slow down the interaction with no tangible improvement in accuracy.

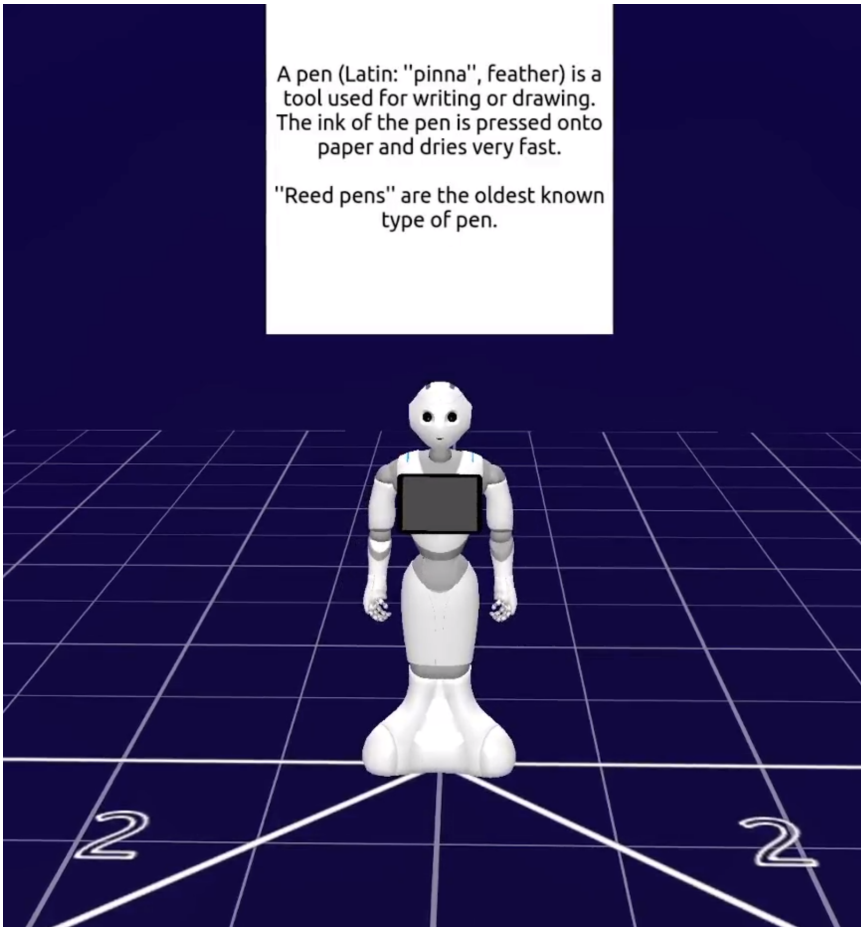

Figure 7: Pepper reads aloud the start of the Simple Wikipedia page of the object that the child has shown him. He will read more or less content from the page depending on the age of the child (more or less than 10 years old).

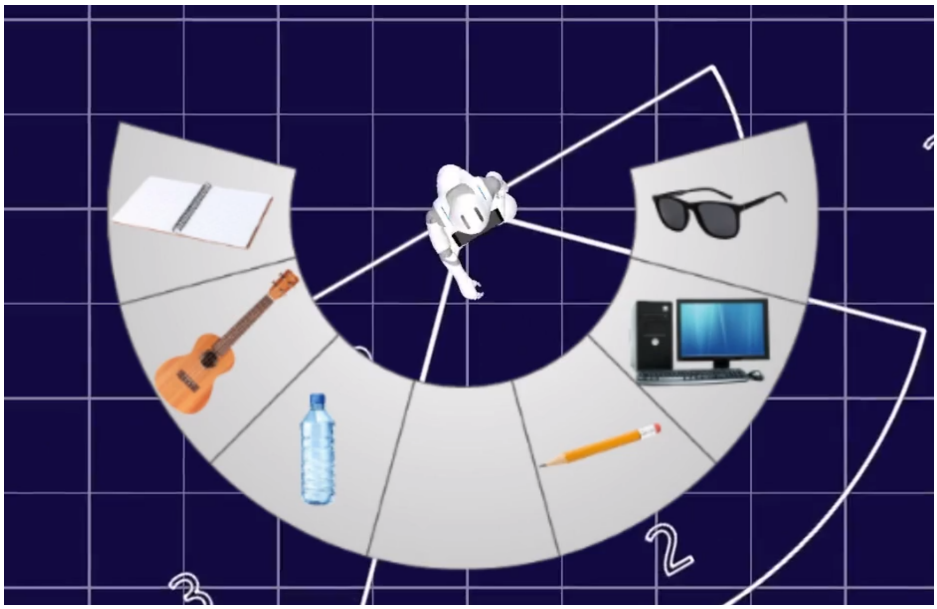

Figure 8: Pepper has been shown a pen before, so now it points to the pencil as it is the object most similar to a pen amongst the ones present on the table.

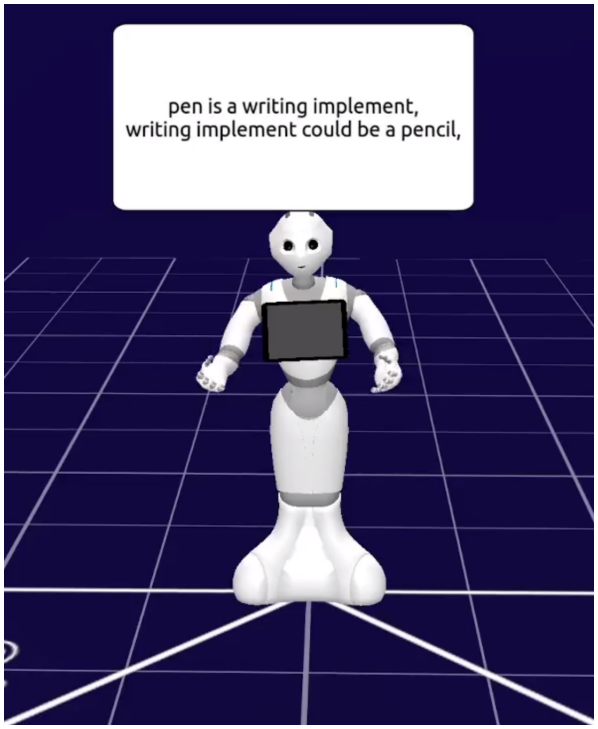

Figure 9: Pepper now uses the A* algorithm to find a path in WordNet between the pen that it has been shown and the pencil on the table (see Figure 12), then the path is converted to the sequence of simple phrases shown in the image via the simple renaming in Table 1

5.2 Interaction with less clear image

In this case, if the confidence of the image recognition is under 95% the robot will ask the child for confirmation about its prediction, in order to avoid errors and at the same time make the experience more interactive for the child.
In fact, it can also ask the child to help it understand what the object is (see Figure 11) by re-taking the photo in better conditions (ex: more light and keeping the object steadier) or just entering the name of the object.

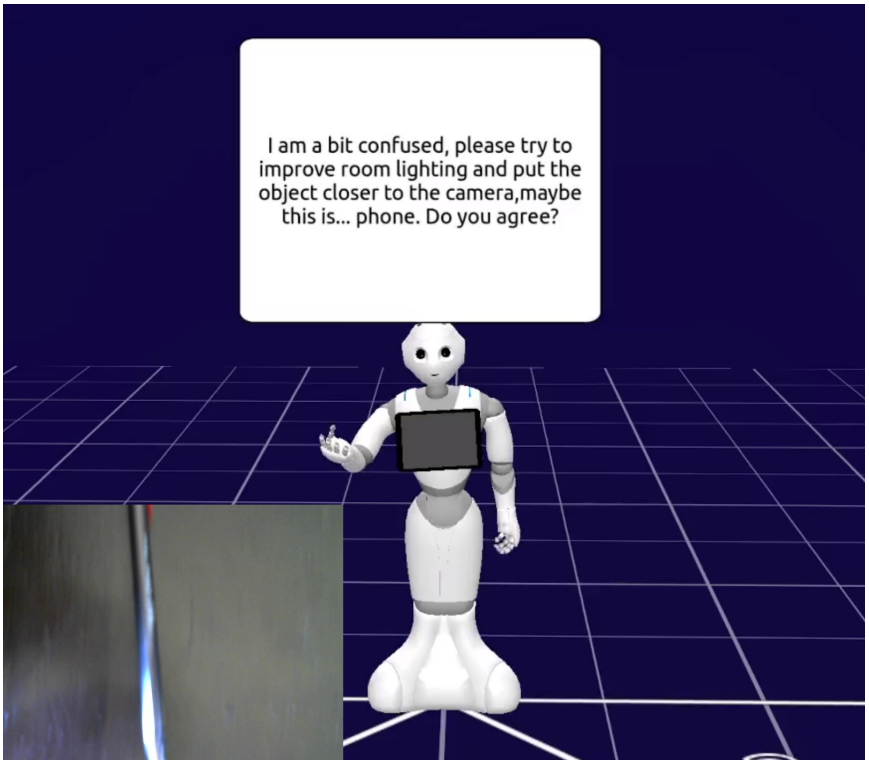

Figure 10: Pepper in this case is not very sure of its prediction ($< 95\%$) so he asks for confirmation in order to avoid errors.

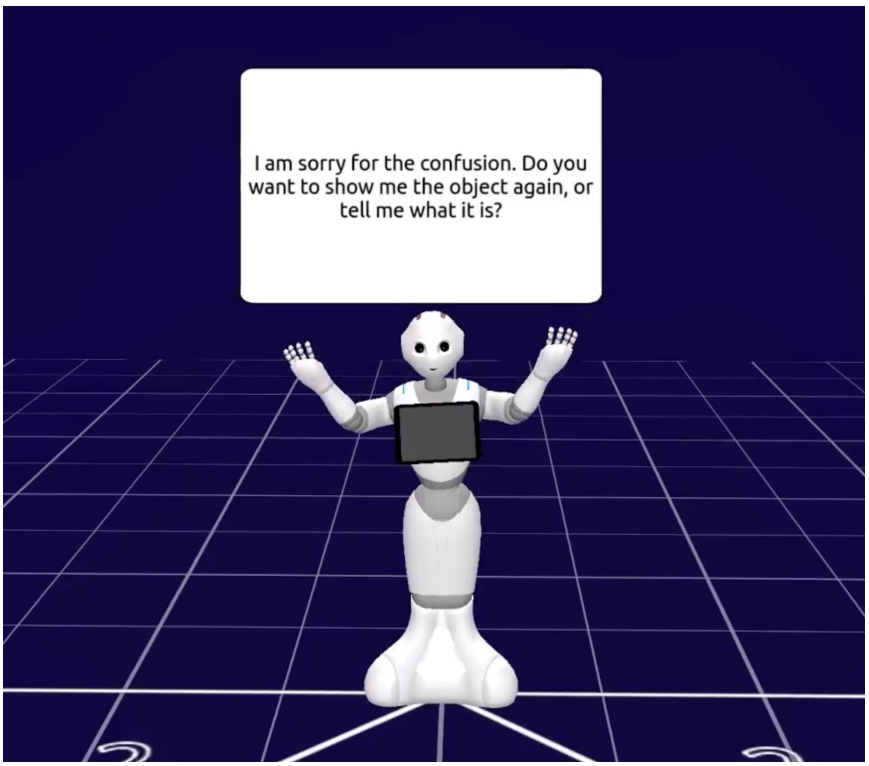

Figure 11: Pepper asks the user to help it further, as it was not able to make a confident prediction (see Figure 12)

.

5.3 Interaction with partial information found

In this case the robot will simply state that the object has not been found in its offline database. We decided to use an offline dump of Simple English Wikipedia for faster lookup, but an online version could also be used in order to be more comprehensive and find more objects at the cost of more complexity, higher latency and the requirement of an internet connection at all times.

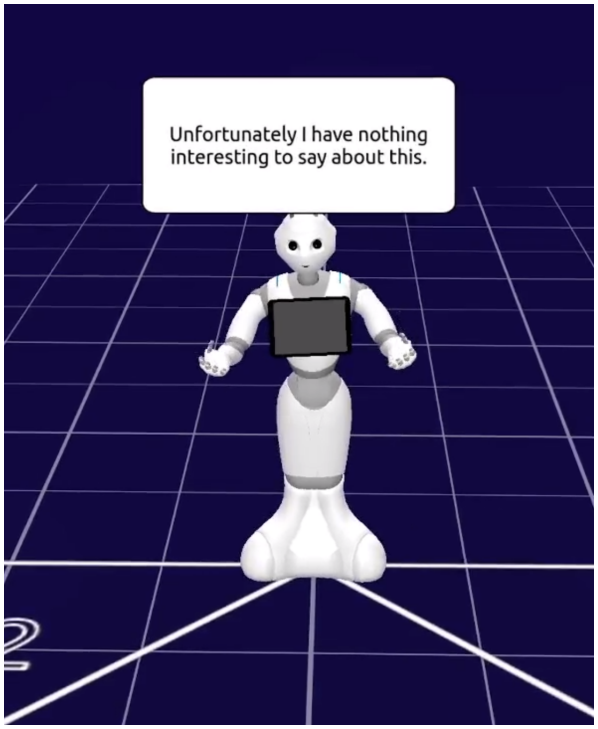

Figure 12: Pepper cannot find information on the object in its offline dump of Simple English Wikipedia, so it just says that it cannot find it and continues with the next step of connecting the object shown with an object on the table.

6 Conclusions

Summing up our project, we have integrated a variety of features: image recognition for the choice of the most similar object, use of the WordNet graph to find an explainable connection, and look-up into Wikipedia to give more information on the object. Our choice of these features is based on the goal of making the interaction satisfying and natural, like when a friend recognizes an article of clothing or an object that we have with us and shows us something similar that he has and explains why they are similar or different. Overall, also by re-watching the final video, we think that all the different functionalities of the robot have been integrated into a coherent whole, making the full interaction with the robot fun and smooth, rather than boring and mechanical.

This project was a great learning experience in integrating a deep learning system (CLIP) together with classical knowledge bases (WordNet and Simple English Wikipedia), and a classical planning algorithm (A*). It has been very instructive trying to find the best way to handle uncertainty in the prediction of the deep learning model and gaps in our knowledge bases. The classical planning algorithm of A* and the WordNet semantic data allowed us to work around the black box nature of the CLIP model by allowing us to give to the child a human-readable explanation of why the object shown and the object on the table are correlated.

A great way to improve this project would be to use a more advanced robot, both on the hardware and software side, in order to be able to pick up and interact with objects in a natural and human-like way. For example, if one of the objects on the table is a notebook or a book with something written on it, the robot could pick it up and read a few phrases from it, if the object is glasses the robot could jokingly put them on, in order to amuse the children, or if the item is a guitar it could pick it up and play a short part of a song. This would require significant hardware and engineering costs, but could be an interesting step towards a more human-like human-robot interaction.

Another possibility is integrating a chat-tuned large language model (such as ChatGPT or an open-source model such as OpenChat) [17, 18] inside the system that receives the Simple English Wikipedia page on the object as input together with questions from the child, in this way, the system will be able to answer questions about the object in a factual way, while also keeping the natural language understanding and fluency of large language models.

A similar result can also be obtained in a less computationally expensive way by dividing the article in paragraphs and using a question-answering transformer sentence-embedding model such as `multi-qa-mpnet-base-dot-v1` [19] in order to retrieve the most relevant paragraph from the article to answer the question asked from the child, this method is both more computationally efficient and guaranteed to be factual, but answering only with sections from the original text can be less fluent.

We look forward to further works in this field as we think that human-robot

interaction for children learning is quite understudied and significant gains and improvements are left to discover, especially given the many promising experiment so far. [10]

References

[1] Weiss García. The teacher shortage is real, large and growing, and worse than we thought. *Economic Policy Institute*, 03 2019.

[2] Aditya Singh and Jaison A. Manjaly. Using curiosity to improve learning outcomes in schools. *SAGE Open*, 12(1):21582440211069392, 2022.

[3] Ali Ayub, Marcus Scheunemann, Christoforos Mavrogiannis, Jimin Rhim, Kerstin Dautenhahn, Chrystopher L. Nehaniv, Verena V. Hafner, and Daniel Polani. Robot curiosity in human-robot interaction (rchri). In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1231–1234, 2022.

[4] Amit Kumar Pandey and Rodolphe Gelin. A mass-produced sociable humanoid robot: Pepper: The first machine of its kind. *IEEE Robotics Automation Magazine*, 25(3):40–48, 2018.

[5] Masahiro Mori and Karl F. Macdorman. The uncanny valley: The original essay by masahiro mori-ieee spectrum. 2017.

[6] George A. Miller. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[7] Zhen-Jia You; Chi-Yuh Shen; Chih-Wei Chang; Baw-Jhiune Liu; Gwo-Dong Chen. A robot as a teaching assistant in an english class. *IEEE*, -(-):–, 2006.

[8] Mitsuharu Matsumoto. Fragile robot: The fragility of robots induces user attachment to robots. *International Journal of Mechanical Engineering and Robotics Research*, 10:536–541, 01 2021.

[9] James Kennedy, Paul Baxter, and Tony Belpaeme. The robot who tried too hard: Social behaviour of a robot tutor can negatively affect child learning. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, page 67–74, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *Science Robotics*, 3, 2018.

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[12] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[13] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. 14:253–302, 2001.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] Xml dump of simple english wikipedia.

[16] Simona Perri Torsten Schaub Benjamin Kaufmann, Nicola Leone. Grounding and solving in answer set programming.

[17] OpenAI. Gpt-4 system card. 2023.

[18] Openchat open source model on github: https://github.com/imoneoi/openchat.

[19] multi-qa-mpnet-base-dot-v1 is a transformer model that maps sentences paragraphs to a 768 dimensional dense vector space and was designed for semantic search. it has been trained on 215m (question, answer) pairs from diverse sources: https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1.