Ontology guided Context Understanding for Robotic Task Execution

Dibyarup Dutta[1], Avigyan Bhattacharya[1], Snehasis Banerjee[2,1]

[1]TCS Research, Tata Consultancy Services, India
[2]Robotics Research Center, IIIT Hyderabad, India

***Abstract*— How can a robot understand the environment it is in, and decide the feasibility of an instructed task in that environment? To address this question, we have developed a system that supports robotic manipulation and navigation by leveraging an extension of IEEE CORA [1] ontology. This ontology contains knowledge of both robotic tasks and entities in an indoor environment. Contrary to the prior works [2] that lacked usage of ontology in task capability understanding, the proposed system uses an explainable semantic approach leveraging perception and semantic web technology to check task execution feasibility in realistic settings. This utilizes scene context understanding or the world model of the robot. Hence, we present a generalized system and method for downstream embodied AI tasks that leverage semantic web techniques combined with perception algorithms. We show that, how using ontology, a robot in exploration mode can create scene graphs that are further used to create a generic knowledge graph and semantic map of the environment to aid the task feasibility analysis. These approaches are tested on realistic simulation environments for later deployment in physical world.**

***Index Terms*— Ontology, Scene Understanding, Task Planning, Cognitive Robotics.**

I. Background

Recently researchers have developed strong interest in benchmark challenge tasks related to Embodied AI [3]. Here, given realistic 3D dataset scenes enabled simulators [4], task specific user instruction and final objective is provided. An example task is 'Object Nav' [5], where given input of text based user instruction like 'go to the dining room', a robotic agent starting at a random location is required to reach in the vicinity of the target location (here dining table) in an unseen indoor scene. The actuation part of this task only contains navigation from start to target location. On the other hand, another instruction like 'bring the apple from dining table' requires a robot to first reach the target location (as a part of ObjectNav task), and then, hand hold and pick the object (apple) using its manipulation and bring back to the user again. So this instruction involves, firstly navigation to target location, secondly manipulation of the object, and finally bringing the object back to user's location (the target now is the user's location). To achieve this, knowledge about the indoor environment, the types of objects it contains, and general relationships of objects with objects and objects with regions, is required to be learnt or known apriori. One way to get this knowledge is to start with an initial knowledgebase and then adapting it based on exploration of similar scenes, and then finally deploying a model for a specific downstream task. In this case, the context of task will be the state of the world model when the robot is getting a task instruction. However, there will be scenarios in which the user instructed task is not feasible to be executed. Here, we further classify this task types represented in an ontology hierarchy as follows:
(1) Feasible Tasks: Tasks that can be done by the robotic agent R in the current context C.
(2) Non-feasible Tasks: Tasks that cannot be carried out by R in C. This can be sub-categorized as follows:
(2.1) Out of vocabulary tasks - these tasks cannot be completed due to lack of any prior semantics matching in the knowledgebase (example, turning on the car engine).
(2.2) Absurd tasks - these tasks cannot be done based on general concepts description (example, put table on cup).
(2.3) Constraint Failing Tasks: each operation of the robotic agent in the physical world has some feasibility checks or constraints in place in the given context. Some examples of task failures are: (a) when the target object is not in this particular indoor layout (b) the robot is asked to do a task that this version and model of the robot is incapable to do due to lack of capability, like, say stirring the sugar water; or climbing stairs for a wheeled robot (c) the target object does not have a matching attribute, like say asking an air-conditioner to control the humidity of the room, but this object model does not support this humidity feature.

We model the above tasks within the framework of OWL ontology to store the general relationships and RDF files to store the facts. For implementation purpose, we have used Protege [6] for initial ontology development. A domain specific ontology is built on top of IEEE CORA [1], OntoScene [7] and SemNav [8] using competency questions [9] specific to concepts relevant to objects, task specifics, robot capability (including sensor capability) and environment constraints. When a user gives a task, just after the task instruction, the system can infer if the task is feasible or not based on semantic search in the knowledgebase, thereby saving time of the robotic agent to avoid dead-end tasks and hinting the user to alter the task instruction towards a feasible task. The following sections provides details on the two modules of scene understanding and task feasibility analysis. The main contributions of this paper are:
(a) an extended ontology is developed for context understanding and task feasibility analysis.
(b) a system for scene representation and task execution is presented that checks task feasibility with explanation.

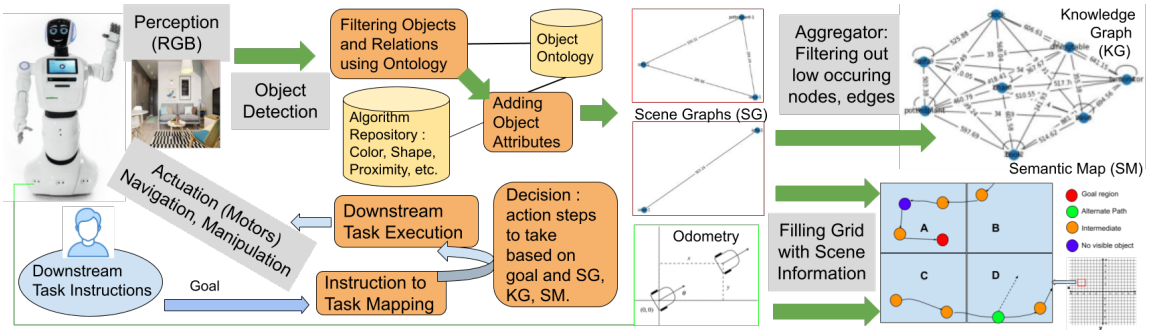

Fig. 1: Ontology Guided Population of Scene Graph, Knowledge Graph and Semantic Map

II. Scene Understanding

Scene graph processing is important for robotic tasks involving camera ego view images to represent symbolically a set of components of the observation as nodes (objects) and edges (inter-node relations). In this regard, Iterative Message Passing [10] took prominence but has the drawback of confusing the order of 'subject-object' in a relationship. Another work [11] finds statistical correlations between object pairs, but ignores less frequent relationships completely, even if they are important. Also, the relations are most often just adjacency, with no scope of further granular or separate attributes with values. We overcome this limitations by using ontology as a reference to generate the scene graph itself.

As shown in Fig. 3, the user gives an instruction for a robotic downstream task (like 'go to the kitchen') that is deciphered into a semantic web based stream of 'subject-predicate-object' conditional states. The robot has a set of continuous inputs at its disposal – RGB camera sensor feeds and wheel odometry readings. The image stream is passed via an image object detector (like YOLO [12] and MaskRCNN [13] variants) to output a set of objects with their respective confidence scores. Next, the pre-built object ontology is queried using SPARQL to fetch object attributes, and a look up is done on the Algorithm repository to check which specific pattern recognition algorithms need to be run on the scene to extract attribute information specific to object(s) in view. The resultant scene graph contains visible objects as nodes and edges as relations. A shape detector is kept to cluster unknown or unclassified objects.

To extract a node properties like colour, 1-Nearest Neighbour approach is used with standard colours representing clusters. For edge property – like proximity with other objects and regions, we compare the location of the bounding boxes and try to normalize them on the basis of weight bands assigned horizontally. Weight banding is needed in cases where the robot camera only has RGB images as input without depth; and in cases where depth perception is beyond the sensitive range of 2.5 meter. We first divide the image equally into number of bands horizontally and then assign weights to each band using a pre-trained model [14] for calculating depth depth maps to establish the proximity relations. Next, the scene graphs across observations are merged to form a single knowledge graph using a similarity based aggregation technique [15] – thereby representing the evolving relationships in the merged graph. The semantic map helps in aligning robot odometry with view angles of robot camera.

Estimating proximity between objects is often challenging as objects detected might not be at the same depth from a RGB camera. We first divide the image equally into number of bands horizontally and then assign weights to each band in the following manner: (a) Using a pre-trained model for calculating depth we find out the depth maps of a certain number of images (b) For each depth map image, we calculate the average of pixel values in a horizontal band (c) Then for each band we calculate average of the band values that we get from all the depth map images i.e., average sum over all images band-wise.

For each object we take the average of the weights in which the corners of the detected bounding box (BB) lie. AS BB is rectangle-shaped in case of YOLO object detection, taking the diagonal points suffice. For example, if any bounding box corner point lies in the height between y1 and y2, it gets assigned its respective weight from the list calculated earlier. We can say that objects at a greater depth will be assigned a lower weight value and objects which are closer get a greater weight value, therefore we are dividing with the weight value so as to normalize them and then calculate the Euclidean distance. Next, a function is mapped from the Euclidean distance (as a reference to pixel distance) to an estimated proximity value in the range of 0 to 1 as an edge attribute (1 means very close).

This knowledge graph expansion approach is tested for navigation and manipulation tasks in the realistic simulation environment of AI2Thor [16]. Simultaneously, a semantic map of the explored area is stored in grid form containing the global map view information in terms of following grid attributes: (a) Target or goal region (b) Alternate paths not visited (c) information regarding observations at that grid cell (d) No objects were detected by perception in that grid area. These two sub-modules: knowledge graph and semantic map is leveraged in the next step of task feasibility analysis.

III. Task Feasibility Analysis

As shown in Fig. 2, the user gives an instruction for a manipulation task that is deciphered into a semantic web based stream of 'subject-predicate-object' conditional states.

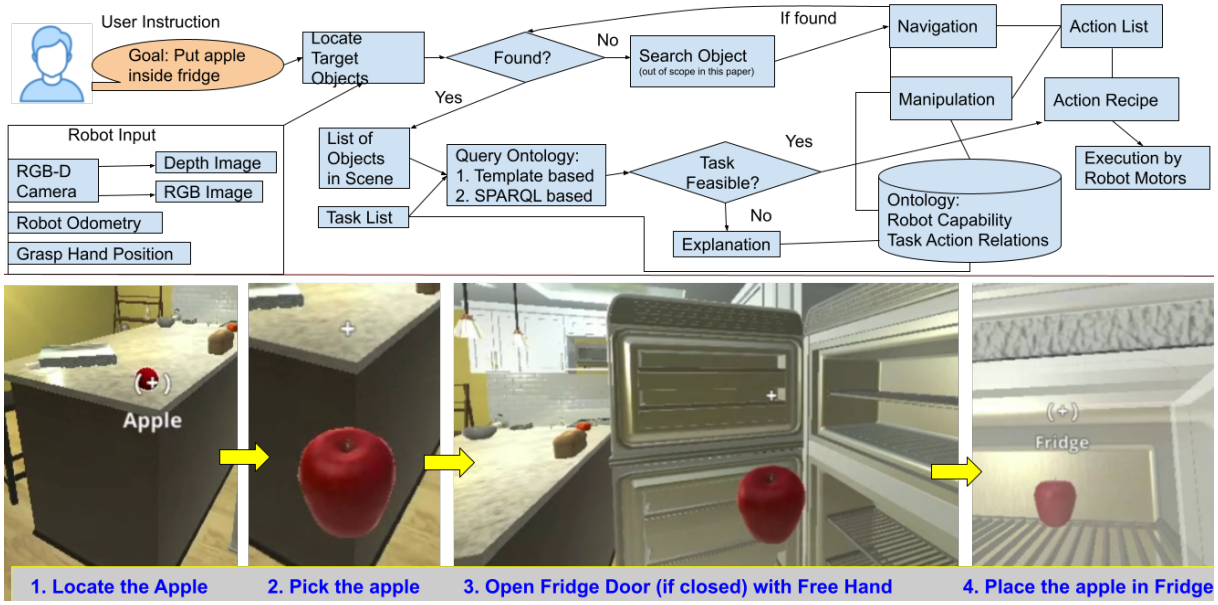

Fig. 2: System for Robotic Manipulation guided by Ontology (with example user instructed task)

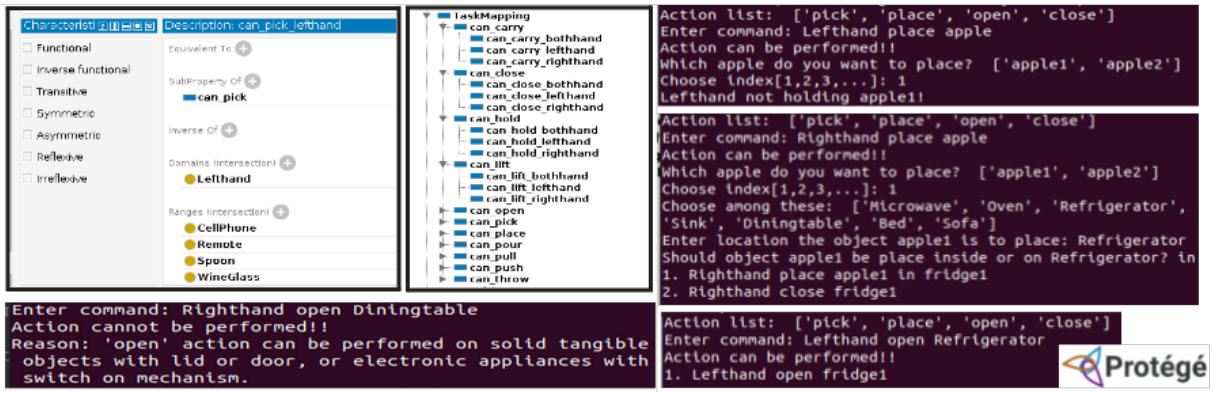

Fig. 3: (a) A view of the ontology (b) Terminal interface for user instructions and robot's responses

The robot has a set of continuous inputs at its disposal, namely, RGB-D or RGB camera sensor, wheel odometry readings, hand grasper co-ordinates and the sensor states. For a given task like placing an apple in the fridge, the system needs to initially check if the target object requiring manipulation is already there in the current field of view. Using the earlier sub-module of scene understanding, the detection is done using standard object detection techniques like YOLO. If object is found to missing or out-of-view, the system needs to invoke ObjectNav task module to 'Search and Find' the object via robot traversal [5], till the object comes in robot's field of view. When the object is in the current view state, the system needs to query the ontology to see a match between the robot capability, target object property and the task type extracted from user instruction. This query can be done either by a set of registered SPARQL queries or by using template based matching. However, template based matching has the drawback of knowing the task type very specifically. In contrast, SPARQL, being a semantic high level query language, enables expression of complex logic more easier in terms of query pattern stacking [17]. The idea is to query the ontology graph's nodes and edge properties to find a pattern match in relation to the predicates of the original task instructed by a user. If the task is not feasible, an explanation will be generated at runtime by combining (a) annotations tagged to specific task failures, and (b) a chain of missing conditions for failing the task, even before start of the task execution. This saves significant task execution time and avoids dead-end situations at runtime.

As an example, if a high level task has five sub-tasks and even one of them, say the last one, is not feasible, the robot should not waste effort to carry out the initial four sub-tasks and then ending up in a state of non-feasible task. This decision should be taken at the start of the task, if possible. If the task is feasible, the robot will look up the task execution list to see which of the sub-task needs to be done, and in what sequence, and involving what entities. The list of task actions and its mapping to actual robotic movements is stored

in Action Recipe module, that acts as a wrapper on top of the actual physical world actuation of a robot. As an example, the high level action 'go forward' will be mapped to moving the robot motor wheels forward by some pre-set distance or instructed distance estimate. Final stage is execution by motors in robotic hand (wheels in case of navigation sub-task) till goal state as per user instruction is reached. Another aspect is the finding of task non-feasibility at the time of task execution due to changes in world. Additionally, failures can happen at the final actuation cycle as robots are physical objects prone to: (a) moving motor parts setbacks and (b) physical sensor errors. Another example follows. Suppose, the robot at the start time found that the task given in Fig. 2 is feasible. But at the last step, it finds the fridge door as locked; and hence is unable to place the object 'apple' inside the fridge. This information of locked door state of fridge is an observation during actuation and was not already known to the robot before the task instruction was given. The system in such a case, can search for a solution by a SPARQL query to list down most likely solutions – like calling a human in the location to unlock the fridge. The solution can be of two types: (a) needs human to solve one or more sub-task step so that the robot can continue with the task execution (b) the robot can take alternate branch path for task execution (c) the robot can query the knowledgebase with current context to get a fresh set of task action sequence which is feasible and will take the robot out of this dead-end. Once a task is found to be infeasible, the reason for the task halt is gathered in terms of a SPARQL query to find the missing links between the goal state of the instruction and the current context scene. A sample explanation run is shown in Fig. 3.

The workflow in Fig. 2 is further explained here. The user instruction gets converted into a set of triples. Each of the predicates of Task List will have some linking to sub-tasks to execute; for example, the 'Move' task for object 'cup' can be broken into: (a) Find cup (b) Pick cup (c) Find table (d) Place cup on table – these are combinations of navigation and manipulation – which will be handled by task executor Action Recipe that maps high level actions to physical world actuation. This abstraction of Action Recipe helps the system to adapt to any robotic system by just having a wrapper function on top of hardware implementation. When target object is in view, the next step is to understand the state of the world by observing other objects which are also in view. Although an object might be visible, but if no path from the current location exists due to some obstacles, the robot needs to find a new unseen free path or tag the task as a failure.

There are different types of task failure cases. As an example, if the instruction is to keep a table on a cup, a SPARQL query will check if the task is possible or not. If the semantics are incorrect, then the task should be stopped. This is type 1 task feasibility failure. An example query is: [*select ?relation from KB where* ⟨*object:table*⟩ ⟨*predicate:LocatedOnTopOf*⟩ ⟨*object:cup*⟩]. If the result of the above query is null, it means that the task is infeasible.

The Type 2 task feasibility failure deals with static information of the world – that includes the capabilities of the robot and the environment settings. An example is asking a robot without an arm to pick an object. In this case, instead of the general common sense based robotic ontology, a specific fact file for the current context needs to be accessed. Robot specific knowledge will get populated based on robot model. A discussed in section 2, The outer world of the robot is generated as a semantic map out of the continuous observations. Another aspect is the presence or absence of static objects in the current scene. Once the robot explores the entire scene, it will have a world map of the environment. Next, if it is asked to execute a task like "check if TV is powered on", but the TV itself is not there in the space, then although the instruction is valid, but it is infeasible in this environment. An example query is of the form: [*select ?object from SemanticMapKB where* ⟨*object:table*⟩ ⟨*predicate:isPresentInWorld*⟩ ⟨*object:SemanticMap*⟩]

The third aspect is Type 3 task feasibility checks, where the dynamic pre-conditions of the task are not satisfied. An example is when the robot is asked to pick an object cup in its arm, when it is already holding a bottle. In this case, the robot is capable of picking it up, but due to the current state of the robot hand, it is unable to do so. An example query is: [*select ?state from SemanticMapKB where* ⟨*object:robot*⟩ ⟨*predicate:hasPart*⟩ ⟨*object:arm*⟩ . ⟨*object:arm*⟩ ⟨*predicate:hasState*⟩ *?state* . *?state* ⟨*predicate:equals*⟩ *'free'^xsd:string*].

The fourth aspect or Type 4 feasibility checks is when while executing the task, some conditions leading to task failure happens which are not pre-anticipated or observed beforehand. An example is trying to put a cup in a fridge whose door is locked. This is observed when the task is commanded for carrying out but robot is not executing it. An example query is: [*select ?state from SemanticMapKB where* ⟨*object:fridge*⟩ ⟨*predicate:hasState*⟩ *'locked'^xsd:string*].

When all the task feasibility checks pass, then only the actual navigation or manipulation takes place by invoking the Action Recipe module as stated earlier. As seen from Fig. 3, this approach is tested in a simulation environment – AI2Thor, that has realistic scenes and physics based objects. In the figure, task action recipe generation is tested on four tasks: pick, place, open and close – this can be extended to further granular and higher level tasks.

IV. Conclusion and Future Work

In this paper, a system for handling robotic context and task feasibility based on an extended ontology is presented. This aids the robot to save time and power by taking decisions before start of an assigned task. An area of future work is processing the perception data in triple format using windowing mechanisms [18] of stream reasoning [19] [20] to handle information in the streaming camera images. In future, we plan to integrate external knowledgebase like DBPedia [21] in the knowledge graph generation stage to get a richer semantic understanding and adaption to new task types. Also we plan to enhance the explanation module by leveraging Large Language Models (LLM) [22].

REFERENCES

[1] E. Prestes, J. L. Carbonera, S. R. Fiorini, V. A. Jorge, M. Abel, R. Madhavan, A. Locoro, P. Goncalves, M. E. Barreto, M. Habib, *et al.*, "Towards a core ontology for robotics and automation," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1193–1204, 2013.

[2] A. Vassiliades, N. Bassiliades, F. Gouidis, and T. Patkos, "A knowledge retrieval framework for household objects and actions with external knowledge," in *International Conference on Semantic Systems*, pp. 36–52, Springer, Cham, 2020.

[3] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 230–244, 2022.

[4] D. P. Kaur, N. P. Singh, and B. Banerjee, "A review of platforms for simulating embodied agents in 3d virtual environments," *Artificial Intelligence Review*, pp. 1–43, 2022.

[5] S. Banerjee, B. Bhowmick, and R. D. Roychoudhury, "Object goal navigation based on semantics and rgb ego view," *arXiv preprint arXiv:2210.11543*, 2022.

[6] T. Tudorache, N. F. Noy, S. Tu, and M. A. Musen, "Supporting collaborative ontology development in protégé," in *The Semantic Web-ISWC 2008: 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings 7*, pp. 17–32, Springer, 2008.

[7] S. Banerjee, P. Pramanick, C. Sarkar, and B. Purushothaman, "Ontoscene: Ontology guided indoor scene understanding for cognitive robotic tasks.," in *ISWC (Industry)*, 2021.

[8] S. Banerjee and B. Purushothaman, "Semnav: How rich semantic knowledge can guide robot navigation in indoor spaces.," in *ISWC (Demos/Industry)*, pp. 398–400, 2020.

[9] C. Bezerra, F. Freitas, and F. Santana, "Evaluating ontologies with competency questions," in *2013 WI-IAT)*, vol. 3, pp. 284–285, IEEE, 2013.

[10] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *CVPR*, pp. 5410–5419, 2017.

[11] T. Chen, W. Yu, R. Chen, and L. Lin, "Knowledge-embedded routing network for scene graph generation," in *CVPR*, pp. 6163–6171, 2019.

[12] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: Challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, pp. 1–33, 2022.

[13] T. Cheng, X. Wang, L. Huang, and W. Liu, "Boundary-preserving mask r-cnn," in *Computer Vision–ECCV 2020*, pp. 660–676, Springer, 2020.

[14] N. U. Islam and J. Park, "Depth estimation from a single rgb image using fine-tuned generative adversarial network," *IEEE Access*, vol. 9, pp. 32781–32794, 2021.

[15] H. L. Nguyen, D. T. Vu, and J. J. Jung, "Knowledge graph fusion for smart systems: A survey," *Information Fusion*, vol. 61, pp. 56–70, 2020.

[16] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, *et al.*, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[17] S. Banerjee and D. Mukherjee, "System and method for executing a sparql query," Feb. 20 2018. US Patent 9,898,502.

[18] S. Banerjee and D. Mukherjee, "Windowing mechanisms for web scale stream reasoning," in *Proceedings of the 4th international workshop on Web-scale knowledge representation retrieval and reasoning*, pp. 17–18, 2013.

[19] D. Mukherjee, S. Banerjee, and P. Misra, "Towards efficient stream reasoning," in *On the Move to Meaningful Internet Systems: OTM 2013 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, ACM, EI2N, ISDE, META4eS, ORM, SeDeS, SINCOM, SMS, and SOMOCO 2013, Graz, Austria, September 9-13, 2013, Proceedings*, pp. 735–738, Springer, 2013.

[20] D. Mukherjee, P. Misra, and S. Banerjee, "System and a method for reasoning and running continuous queries over data streams," June 5 2018. US Patent 9,990,403.

[21] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[22] C. Zhang, C. Zhang, S. Zheng, Y. Qiao, C. Li, M. Zhang, S. K. Dam, C. M. Thwal, Y. L. Tun, L. L. Huy, *et al.*, "A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need?," *arXiv preprint arXiv:2303.11717*, 2023.