Memory Net: Generalizable Common-Sense Reasoning over Real-World Actions and Objects

Julian Eggert, Jörg Deigmöller, Pavel Smirnov, Johane Takeuchi, Andreas Richter

2023

Memory Net: Generalizable Common-Sense Reasoning over Real-World Actions and Objects

Julian Eggert[1], Joerg Deigmoeller[1], Pavel Smirnov[1], Johane Takeuchi[2] and Andreas Richter[1]
[1]*Honda Research Institute Europe, Carl-Legien-Straße 30, 63073 Offenbach am Main, Germany*
[2]*Honda Research Institute Japan, 8-1 Honcho, Wako, Saitama 351-0114, Japan*
{*first name.last name*}*@honda-ri.de, johane.takeuchi@jp.honda-ri.com*

Keywords: Situational Question Answering, Knowledge Representation and Reasoning, Natural Language Understanding

Abstract: In this paper, we explore how artificial agents (AAs) can understand and reason about so called ”action patterns” within real-world settings. Essentially, we want AAs to determine which tools fit specific actions, and which actions can be executed with certain tools, objects or agents, based on real-world situations. To achieve this, we utilize a comprehensive Knowledge Graph, called ”Memory Net” filled with interconnected everyday concepts, common actions, and environmental data. Our approach involves an inference technique that harnesses semantic proximity through subgraph matching. Comparing our approach against human responses and a state-of-the-art natural language model based machine learning approach in a home scenario, our Knowledge Graph method demonstrated strong generalization capabilities, suggesting its promise in dynamic, incremental and interactive real world settings.

1 INTRODUCTION

Our work aims to enable artificial agents (AA) to use practical, everyday common-sense knowledge in physical environments. The AA should evaluate actions, tools, objects, locations, and collaboration partners within a specific context. We use a simulated home environment paired with a Knowledge Graph, known as Memory Net (MemNet). A natural language interface aids in aligning text input with MemNet concepts. This concept aligns with Embodied Question Answering (EQA), (Das et al., 2017), but emphasizes situational context, hence termed Situational Question Answering (SQA), (Deigmöller et al., 2022).

We contend that SQA is important for effective human-machine cooperation, rooted in mutual situational understanding. This shared comprehension lets agents predict or strategize, advancing towards Cooperative Intelligence (Sendhoff and Wersing, 2020), which promotes harmonious human-machine task collaboration.

In this study, we gauge mutual understanding through a question-answering task. The AA, using MemNet, suggests solutions, ranking them from most to least suitable. In a tool identification task, the AA compares human-selected tools to those from MemNet for questions like ”What can I use to store a book?”. The choices are limited to 69 tools within the simulated environment.

All common-sense and instance data exist in a single graph, ensuring streamlined reasoning. We've equipped MemNet with an inference system utilizing semantic proximity via approximate subgraph matching over action patterns.

The paper will first investigate related works, followed by explaining the principles of MemNet, and finally compare our system's performance against human annotated data as well as a neural language model in the tool identification task.

1.1 Related Work

Artificial agents (AA) rely heavily on the quality of their knowledge sources for question-answering performance. Knowledge-graphs like WikiData and DBPedia predominantly offer factoid, encyclopedic knowledge about various topics (Antoniou and Bassiliades, 2022). Systems built on such data often pair natural language understanding with SparQL query generation (Zafar et al., 2018; Diefenbach et al., 2018; Diefenbach et al., 2020), with benchmarks centered around instance-related questions (Dubey et al., 2019). While these systems show progress, relying solely on encyclopedic knowledge isn't sufficient for embodied agents to interact in a physical environment.

Common-sense knowledge graphs, unlike encyclopedic ones, focus on general world knowledge. These are sourced from plain texts, search-engine logs, language models, or human curations such as WebChild (Tandon et al., 2017), Atomic (Sap et al., 2019), and Wordnet (Miller, 1998). Systems based on this knowledge usually employ neural networks or language models for question-answering (Storks et al., 2019).

Embodied Question Answering (EQA) handles questions about physical entities, necessitating both natural language understanding and robot localization and navigation. However, benchmarks like (Shridhar et al., 2020; Srivastava et al., 2022) mainly assess sequences of expert-driven actions rather than the robot's common-sense capabilities. (Vassiliades et al., 2021) explored richer semantic understanding of scene objects and implemented a knowledge retrieval system for cognitive robots. Knowledge representations in robotics, like (Paulius and Sun, 2018) and (Thosar et al., 2018), detail actions for task executions but often overlook language interactions, especially in resolving ambiguities. While some robotics applications include language understanding, they typically lack the extensive knowledge needed to execute complex tasks or comprehend their context (Fischer et al., 2018; Venkatesh et al., 2021; Lynch and Sermanet, 2021; Mühlig et al., 2020).

A promising development is found in (Ahn et al., 2022), which employs a large language model for common-sense reasoning. They introduced an affordance function to anchor the language model in the physical world. However, we advocate for structured knowledge representation for transparency in decision-making and real-time knowledge incorporation, especially if robot behaviors or knowledge need refining through interactions.

2 MEMORY NET KNOWLEDGE GRAPH

We utilize a unique knowledge graph structure for common-sense reasoning, as presented in (Eggert et al., 2020) and (Eggert et al., 2019). This graph describes heavily interlinked entities, with nodes representing concepts and instances, and links denoting relationships between them.

Our knowledge graph acts as the sole storage for varied data: from abstract concepts to real-world entities, objects, tools, actions, states, and even temporal observations. Unlike the semantic web (Berners-Lee et al., 2001) and many property graphs (Angles, 2018) that rely on node and link category naming to carry semantics requiring human interpretation, our MemNet assumes no implicit semantics. This ensures entirely machine-driven operations and allows the AA to expand its knowledge during operations.

Concept semantics emerge from the graph's structure and a minimal set of link types, enabling operations like property reasoning (”hasProp”), compositionality (”hasPart”), and transformations (”transTo”). In MemNet, a ”property pattern” represents each concept, linking a central node (the main concept) to child nodes that signify subconcepts and properties. These properties ideally correlate to external ”measurements”, such as linguistic utterances or visual recognition.

Except for certain root nodes, all concept nodes are automatically linked to others via specialization or inheritance links. A concept specializes from another if it maintains its parent's traits but has additional unique properties and relationships. This specialization is explicitly denoted in the graph, helping to maintain a concise representation and minimize redundancies. The concept's meaning arises both from its property pattern and its links to other concepts.

One central concept of an AA that operates in real-world is that of an action, since it links multiple different types of concepts, such as involved objects and tools, a predicted result or a participating actor. This is closely tied to verb semantics as highlighted in linguistics (Baker et al., 1998).

Within our knowledge graph, an ”action pattern” defines the components of an action. These are specialized versions of original concepts, given their specific roles in the action. Hierarchical taxonomies help detail agents, transformations, and objects, which then take on specific roles in an action. For instance, a knife becomes a tool in the context of the ”cut” action, while an agent might be an actor or recipient in a ”bring” action, depending on its context.

A key benefit of MemNet's specialization schema is its seamless representation of both abstract concepts and real-world entities.

3 EVERYDAY COMMON-SENSE KNOWLEDGE

To infuse our MemNet with general-knowledge concepts and links, we utilized multiple taxonomies from public datasets. Primarily, we turned to WordNet (Miller, 1998) to extract hub nodes for specialization taxonomies. These nodes, organized as ”synsets”, connect synonyms with shared meanings, serving as our MemNet's foundational structure. However, WordNet lacks details on action patterns.

To fill this gap, we leveraged ConceptNet (Speer et al., 2017). Its natural language triplets, such as ”[a net]-[used for]-[catching fish]”, were connected to MemNet concepts through spaCy[1](Honnibal and Montani, 2017). For accurate connections, action and object lemmas from phrases were extracted (Losing et al., 2021), and Word Sense Disambiguation facilitated the mapping between WordNet synsets and MemNet concepts.

Although ConceptNet was useful, its domain coverage was inconsistent. From it, we retrieved 271 statements, primarily of the ”used for” relation that were relevant for our scenario. To broaden our coverage, we introduced 70 high-level action patterns, examples include:

- make.v.03, food.n.01, cooking_utensil.n.01
- make.v.03, food.n.01, kitchen_appliance.n.01
- drink.v.01, beverage.n.01, drinking_vessel.n.01
- ...

For practical testing, we employed VirtualHome (Puig et al., 2018), simulating an agent’s real-world interactions. We incorporated 69 stateful objects from VirtualHome into our graph.

In summary, three sources: WordNet for base categorizations, ConceptNet for action patterns, and VirtualHome for tangible household items, enriched our MemNet. On top, a language interface was integrated, utilizing syntactic parsing via spaCy, allowing us to transform user queries into MemNet queries. For more information, refer to (Deigmöller et al., 2022).

4 REASONING AND GENERALIZATION

The MemNet database now integrates information on both abstract and everyday concepts, objects, instances, and the pivotal action patterns linking them. Importantly, every concept, object, and action pattern has an associated natural language expression (”lemmas”).

Envision this typical reasoning process: A user poses a query with an incomplete action pattern. Perhaps they’re asking about the object being acted upon, a related tool, its common location, or the acting agent. A hint typically suggests the missing piece, which might be an agent, object, or tool. Questions could be phrased as, ”Which tool can I use to cut onions?” (here, the missing element is a tool) or ”Who can take that?” (where an agent is the missing link).

Addressing communication uncertainties, ambiguities, or general underspecifications is crucial. Our method utilizes the graph structure, transforming the challenge of pinpointing compatible action patterns into an approximate subgraph matching problem—governed by semantic distance. This distance is gauged by the connectivity of the knowledge engine. We’re primarily searching for closely connected action patterns, enabling the system to generalize and tap into the depth of the MemNet graph.

The reasoning process is tri-phasic: association, generalization, and filtering. In the association phase, lemmas guide towards related concepts in our database. In generalization, a broad spectrum of semantically linked action patterns is explored. Finally, these patterns are verified based on a) the semantic distance between their concepts and the initial hints, and b) their appropriateness in the given context, like item availability or achieving a goal state. Prioritizing by semantic distance refines the focus on the most relevant options.

4.1 Action Pattern Inference

In a query such as ”What can I use to cut onions?”, our language understanding identifies ”cut” as the action’s lemma and ”onions” as the object’s lemma. Given the question structure, we infer that a tool is being sought, initializing the tool’s lemma as ”anything”. Since lemmas are tied to concepts, the initial step is identifying all concepts linked with ”cut”, ”onions”, and ”anything”. This identifies our seed concepts for actions, objects, and tools (lines 1-3 in Listing 1). Owing to language’s inherent ambiguity, there might be multiple seed concepts for each lemma.

Next, these seed concepts link to more specific subconcepts and broader superconcepts through specialization links. For instance, ”cutting” can be generalized as ”separating” or ”dividing” and specialized as ”slicing” or ”cubing”. A similar procedure applies to ”onions”. This step enlarges our pool of concept candidates (lines 4-6).

We then identify matching action patterns by ensuring they integrate an action, object, and tool concept, each stemming from the respective set of concept candidates or their specialized versions in the action pattern (line 7).

Finally, line 8 directs us to the action patterns’ originating action, object, and tool concepts from the candidate concept sets. These foundational concepts guide us in identifying real-world instances fit for the action patterns. For example, in the question ”What can I use to cut onions?”, while a real-world object

[1] https://spacy.io/

like a ”knife” is a viable candidate, it isn’t involved in any specified action yet.

Listing 1: Starting from lemmas for action pattern elements like action, object and tool, search the knowledge graph for semantically related action patterns

```
action_seed = Get concept with lemma
     action_lemma
object_seed = Get concept with lemma
     object_lemma
tool_seed = Get concept with lemma tool_lemma
action_concept_candidate = Get sub+superconcepts
      of action_seed
object_concept_candidate = Get sub+superconcepts
      of object_seed
tool_concept_candidate = Get sub+superconcepts
     of tool_seed
(action, object, tool, ...) = Get all AP's where
      action is subconcept from any of
     action_concept_candidate AND object is
     subconcept from any of
     object_concept_candidate AND tool is
     subconcept from any of
     tool_concept_candidate
(action_concept, object_concept, tool_concept,
     ...) = Get corresponding superconcepts
     closest to action, object, tool, ...) from
     the concept candidate pools
```

4.2 Generalization and Ranking

Action patterns, retrieved as per listing 1, match the initial lemmas. Moreover, by associating these with more abstract or specialized concepts, the system can retrieve a broader range of action patterns semantically linked to the user’s query. For instance, asking ”What can I use to cut onions?” might yield results not only specific to ”cutting onions with a knife” but also more generalized or specialized patterns like ”separating vegetables with a cutting utensil” or ”cubing vegetables with a paring knife”, provided they exist in the knowledge base.

Once these patterns are retrieved, their semantic proximity is evaluated. To achieve this, the shortest abstraction or specialization paths between the seed concepts and the concepts currently populating the AP candidates are identified (lines 1-3 in listing 2). These paths then inform the computation of element-wise abstraction or specialization distances (lines 4-9).

For every action pattern, the total semantic distance is computed by summing up their individual element-wise distances (lines 10-11). These distances guide the ranking of the action patterns. Specialization is straightforward since a specialized AP is wholly compatible with a broader one. In contrast, abstracting creates broader APs, which can be less accurate in specific contexts.

The resultant ranking arranges the responses, focusing on the most pertinent ones, typically those closest to the initial query. While highly abstract APs like ”manipulate solid food with a tool” might be valid, they may not be relevant. On the other hand, extremely specialized APs, like ”slicing those shallots with a particular slicer only Tom owns,” might not resonate with the broader context.

Listing 2: Ranking of action patterns by semantic similarity

```
1  action_path = Get shortest sub+superconcept path
      between action_seed and action_concept
2  object_path = Get shortest sub+superconcept path
      between object_seed and object_concept
3  tool_path = Get shortest sub+superconcept path
     between tool_seed and tool_concept
4  action_abstraction_distance = length(action_path
     ) if action_concept is more abstract then
     action_seed, 0 otherwise
5  object_abstraction_distance = length(object_path
     ) if object_concept is more abstract then
     object_seed, 0 otherwise
6  tool_abstraction_distance = 0 or length(
     tool_path) if tool_concept is more abstract
      then tool_seed, 0 otherwise
7  action_specialization_distance = length(
     action_path) if action_concept specialized
     from (action_seed), 0 otherwise
8  object_specialization_distance = length(
     object_path) if object_concept specialized
     from object_seed, 0 otherwise
9  tool_specialization_distance = length(tool_path)
      if tool_concept is specialized from
     tool_seed, 0 otherwise
10 ap_abstraction_distance = weighted sum of (
     action_abstraction_distance,
     object_abstraction_distance,
     tool_abstraction_distance, ...)
11 ap_specialization_distance = weighted sum of (
     action_specialization_distance,
     object_specialization_distance,
     tool_specialization_distance, ...)
```

4.3 Retrieval of Instance Candidates

Once we have a ranked set of compatible action patterns, we can proceed to check if there are some concrete items in the environments that might be compatible with those actions. For this purpose, we search for concepts that are specializations of the action, object, tool, ... concepts of the AP’s and which fulfill certain additional criteria.

Since one constraining criterion is given by the seed information itself (e.g. if the user asks ”What knife can I use to cut onions?”, the category ”knife” of the tool is already given as a seed), we determine the pairwise lowest (in terms of specialization distance) between the seed concepts and the concepts of the AP elements, see lines 1-3 of listing 3. From that starting point, in lines 4-6 we proceed to find specialized versions of those concepts.

Let us concentrate on the case of a requested tool in ”What can I use to cut onions?”. From the request, there is no further constraint on the tool, so the tool seed can be anything that is usable as a tool. One might consider e.g. ”knife”, ”cutting tool”, ”kitchen knife” and ”swiss army knife” as tool concepts being part of corresponding AP's. Searching for instances of these concepts in a real scene might reveal that there is no "swiss army knife", but several instances of kitchen knives and other cutting tools. The semantic distance evaluation finally reveals that a kitchen knife might be the best candidate for using as a tool.

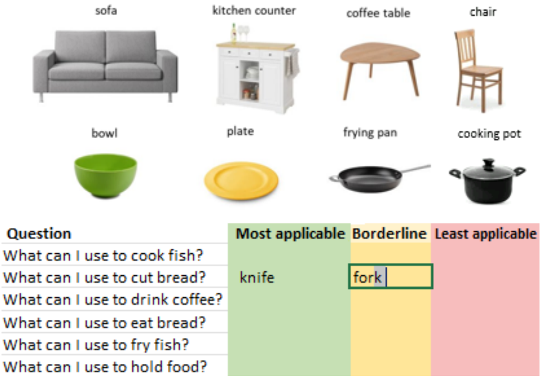
Listing 3: Retrieval of suitable candidates for action pattern elements

```
action_instance_seed = Get pairwise lowest
     concept between action_seed and
     action_concept
object_instance_seed = Get pairwise lowest
     concept between object_seed and
     object_concept
tool_instance_seed = Get pairwise lower concept
     between tool_seed and tool_concept
action_instance_candidate = Get subconcepts from
      action_instance_seed
object_instance_candidate = Get subconcepts from
      object_instance_seed
tool_instance_candidate = Get subconcepts from
     tool_instance_seed
```

5 EXPERIMENTAL RESULTS

To evaluate the reasoning and generalization performance of the knowledge engine, we compiled a set of 109 questions related to our virtual home environment and asked 20 subjects to answer them. More specifically, we asked questions about which object/-tool that is present in the environment, might be useful to execute a certain action on another specific object, e.g. "What can I use to drink juice?". The subjects could pick any candidate from the list of 47 object/tool types (indicated in Fig. 1), and order them from most to least applicable. The constraint was to provide at least 3 candidates for each question. For the benchmark, we collected the votes for each candidate, finally resulting in an ordered answer set for each question ranked from high to low number of votes. For the benchmark, answers with less than 3 votes have been excluded. The goal of the evaluation is to find out: (1) if the algorithm correctly identifies, i.e. disambiguates the right action-object-tool context from lemmas; (2) how well the results match with the expectations of the human subjects. We could also phrase this as identifying the shared context understanding between human and machine, as stated in the section 1.

To judge the knowledge engine performance, we compared two more approaches with the user answers. One is based on a fine-tuned neural language model and the other provides an as simple as possible baseline. Both are explained below in further detail.



Question	Most applicable	Borderline	Least applicable
What can I use to cook fish?			
What can I use to cut bread?	knife	fork	
What can I use to drink coffee?			
What can I use to eat bread?			
What can I use to fry fish?			
What can I use to hold food?			

Figure 1: Excerpt of the tool icons (top) and an excerpt of the questionnaire (bottom). The questions refer to tools available as tool icons that could be applied for a given question stating an action and object, like "What can I use to drink coffee?". The subjects had to select at least three possible answer candidates from the tool icons, which correspond to the objects available in our simulator scene. The candidates had to be assigned to one of the categories from preferred (green) to least applicable (red).

5.1 Baseline

To generate a baseline answer set, we used the distribution of the human answers. Based on the histogram over all answers for all questions, we sorted the tool candidates by the number of votes. This ranked order is used as standard answer set to every question. That means, we always give the same answer, independent of the question, based on the frequency of the candidates in the overall user answers. Even if this approach is simple, it is a quite reasonable strategy in terms of choosing a correct answer if the system is agnostic of the question, because our set of answer candidates, i.e. instances, is limited.

5.2 Neural Language Model

The Neural Language Model (NLM) we incorporated is rooted in deep learning. We trained and tested the NLM as a binary classification problem so that the NLM could deal with a small amount of training data.

The dataset employed to train the NLM was sourced from the same set of information archived in MemNet, detailed in Section 3. Leveraging triples from ConceptNet such as [bookshelf]-[used for]-[storing books] or their equivalent representations in MemNet's action pattern, we autonomously spawned a question-response duo: "What can I use to store a book?" paired with the answer "bookshelf". Consequently, this provided us with one labeled instance for supervised training, tagged with the output "1". Similarly, negative samples for this question statement are generated by randomly selecting from other tool can-

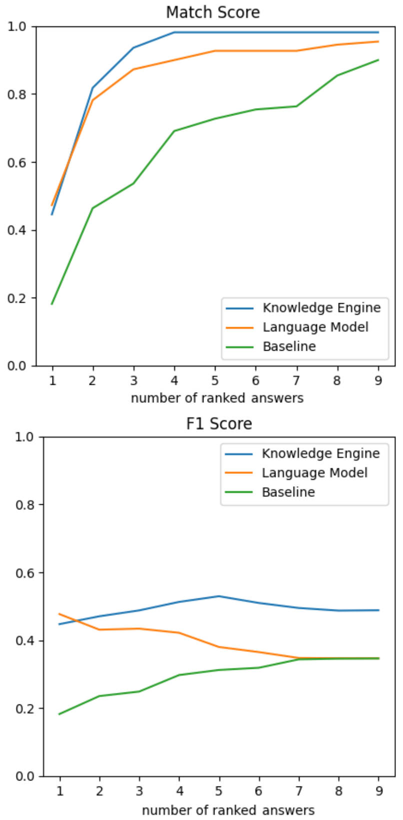


Figure 2: Match score (top) and F1 score (bottom) for increasing number of answers, comparing three approaches, the knowledge engine (MemNet), the language model and the baseline.

didates, like e.g. ”glass” is chosen, which then gets an output value of 0.

The model was constructed by fine-tuning with the above training data using BERT (Devlin et al., 2018). For each positive sample, $\sim$20 negative examples were generated, with a total number of 6030 negative examples.

Upon completion of the training phase, the refined neural model produced a value within the range $[0,1]$ for every question-tool combination. This value was interpreted as the likelihood of the statement being accurate. For every query posed, we then arranged all plausible tool responses based on this computed likelihood. This provided a hierarchical list of potential responses, facilitating a side-by-side comparison with the datasets generated by human participants and the MemNet.

5.3 Evaluation Measures

For our evaluation, we have chosen two measures: the standard F1 score and a match score explained in more details below. We introduced a match score measure because of two reasons. First, we cannot map standard question answering measures, where the most obvious is the mean reciprocal rank (MRR, (Möller et al., 2022)) to our problem setting. This is because we have a set of reference answers and a set of system answers, instead of only a single reference answer as it is used for the MRR. Second, we think that the match score reflects well the message of our investigation, which is how does the tool candidates a human has in mind for a given question overlap with the set of system responses.

The match score is motivated by set theory and is based on the intersection $R \cap S$ between the reference answer set R and the system answer set S. Further, we count a positive match if there is any intersection between the two sets by $|R \cap S| > 0$, otherwise we count it as a no match by:

$$M = \begin{cases} 1 & \text{if } |R \cap S| > 0 \\ 0 & \text{otherwise} \end{cases}$$

To analyze the ranking of answers, we compute M for an increasing set size of R and S between 1 and 10 (cf. Fig. 2).

Similar to the match score, we can use the sets for calculating the F1 score, where precision = $|R \cap S|/|S|$ and recall = $|R \cap S|/|R|$. Again, we compute the F1 score for increasing set sizes, which gives us the performance curve (cf. Fig. 2 bottom).

5.4 Comparison of Results

We compared three approaches against our reference answers from the subjects. The match score at Fig. 2 (top) shows that the baseline method performs worst, far below the language model and the knowledge engine. Nevertheless, the answers provide already a quite good guess, even if it is not related to the individual question.

The language model and the knowledge engine perform comparably well considering the first 2 answers only. If we increase our set of answers to 3 and higher, we can see that the knowledge engine outperforms the language model, with a maximum difference of around 10%. With increasing set sizes, all approaches converge for the match score, which means that the set size is large enough to cover at least one correct answer in the reference data.

The plots in Fig. 3 show how many candidates contribute to a positive match for the knowledge engine (top) and the language model (bottom). The distribution indicates that the knowledge engine provides significantly more correct answers in the earlier ranks than the language model. This is especially the case for matches including 4 (red) and 5 (purple) correct

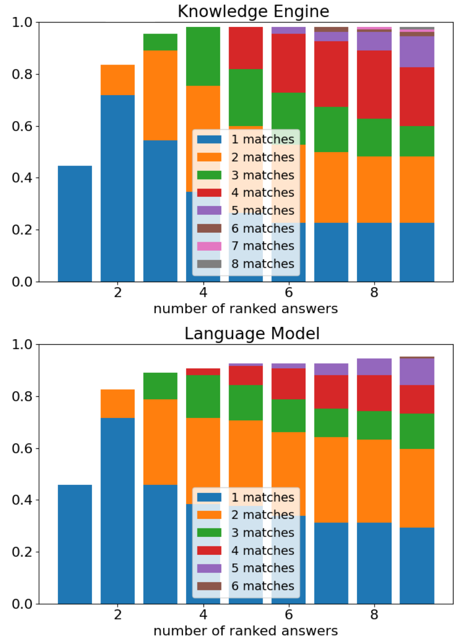


Figure 3: Number of correct answers for knowledge engine (top) and language model (bottom). It shows the number of correct candidates for increasing number of included answers, as in Fig. 2. The envelope matches with the Match Score from Fig. 2.

candidates.

Looking at the F1 score in Fig. 2 (bottom), we can observe a similar tendency as for the match score (top). The difference between knowledge engine and language model gets even more prominent. Probably, this is due to the difference in the number of correct candidates per set, which we already observed in Fig. 3.

6 CONCLUSION

In this paper, we introduced a generalizable inference method for Memory Nets — a type of knowledge graph — that reasons about actions using action patterns. These patterns, derived from common-sense facts and sources like ConceptNet to connect objects, tools, agents, or locations in an action context.

We assessed this method by comparing user and system responses to scenario-based tool-related questions. Alongside, a language model trained on the same common-sense facts was used for benchmarking. Our findings from section 5.4 highlight that our approach offers similar or better inference than the language model. Notably, our method provides more rational answers by working at a conceptual level, while the language model hinges on word patterns. A significant advantage of our system is its transparent reasoning through the knowledge graph.

However, we aim beyond just comparison. Conventional machine learning models, like BERT, demand retraining for each unique task or situation. While the underlying BERT language model might contain latent representations about objects, tools and actions from language correlation statistics, the embedding of the system into a specific situational context occurs by the training data, meaning that the system has to be retrained for every task and every situation. In contrast, Memory Nets assimilate changing situations or new action patterns without retraining, thanks to their graph-based inference mechanism.

As a conclusion, we believe that a system which operates in dynamically changing situations and which deals with an incrementally growing knowledge base will not be possible with standard batch-learning based models but will require more flexible reasoning mechanisms like proposed in this paper. Anyhow, targeting for a hybrid solution, combining advantages of both approaches, is a promising direction.

REFERENCES

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Angles, R. (2018). The property graph database model. In *AMW*.

Antoniou, C. and Bassiliades, N. (2022). A survey on semantic question answering systems. *The Knowledge Engineering Review*, 37.

Baker, C., Fillmore, C., and Lowe, J. (1998). The berkeley framenet project. In *Proceedings of the Coling-Acl*.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific american*, 284(5):34–43.

Das, A., Datta, S., Gkioxari, G., S. Lee, D. P., and Batra, D. (2017). Embodied question answering. https://arxiv.org/abs/1711.11543. Accessed: 2022-12-16.

Deigmöller, J., Smirnov, P., Wang, C., Takeuchi, J., and Eggert, J. (2022). Situational question answering using memory nets. In *14th International Conference on Knowledge Engineering and Ontology Development(KEOD)*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diefenbach, D., Both, A., Singh, K., and Maret, P. (2018). Towards a question answering system over the semantic web (2018). *arXiv preprint arXiv:1803.00832*.

Diefenbach, D., Giménez-García, J., Both, A., Singh, K., and Maret, P. (2020). Qanswer kg: designing a portable question answering system over rdf data. In *European Semantic Web Conference*, pages 429–445. Springer.

Dubey, M., Banerjee, D., Abdelkawi, A., and Lehmann, J. (2019). Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International semantic web conference*, pages 69–78. Springer.

Eggert, J., Deigmöller, J., Fischer, L., and Richter, A. (2019). Memory nets: Knowledge representation for intelligent agent operations in real world. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 260–282. Springer.

Eggert, J., Deigmöller, J., Fischer, L., and Richter, A. (2020). Action representation for intelligent agents using memory nets. In *Communications in Computer and Information Science*, volume 1297. Springer.

Fischer, L., Hasler, S., Deigmöller, J., Schnürer, T., Redert, M., Pluntke, U., Nagel, K., Senzel, C., Ploennigs, J., Richter, A., et al. (2018). Which tool to use? grounded reasoning in everyday environments with assistant robots. In *CogRob@ KR*, pages 3–10.

Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://sentometrics-research.com/publication/72. Accessed: 2022-12-16.

Losing, V., Fischer, L., and Deigmöller, J. (2021). Extraction of common-sense relations from procedural task instructions using bert. In *Proceedings of the 11th Global Wordnet Conference*, pages 81–90.

Lynch, C. and Sermanet, P. (2021). Translating natural language instructions to computer programs for robot manipulation. In *Robotics Science and Systems*.

Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

Möller, T., Pietsch, M., and Rusic, M. (2022). Metrics to evaluate a question answering system. https://www.deepset.ai/blog/metrics-to-evaluate-a-question-answering-system. Accessed: 2022-12-16.

Mühlig, M., Fischer, L., Hasler, S., and Deigmöller, J. (2020). A knowledge-based multi-entity and cooperative system architecture. In *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, pages 1–6. IEEE.

Paulius, D. and Sun, Y. (2018). A survey of knowledge representation in service robotics. In *Robotics and Autonomous Systems*.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. (2018). Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., and Choi, Y. (2019). Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3027–3035.

Sendhoff, B. and Wersing, H. (2020). Cooperative intelligence – a humane perspective. In *2020 IEEE Conference on Human-Machine Systems*.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Speer, R., Chin, J., and Havasi, C. (2017). Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu, K., et al. (2022). Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR.

Storks, S., Gao, Q., and Chai, J. Y. (2019). Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*, pages 1–60.

Tandon, N., De Melo, G., and Weikum, G. (2017). Webchild 2.0: Fine-grained commonsense knowledge distillation. In *Proceedings of ACL 2017, System Demonstrations*, pages 115–120.

Thosar, M., Zug, S., Skaria, A. M., and Jain, A. (2018). A review of knowledge bases for service robots in household environments. In *6th International Workshop on Artificial Intelligence and Cognition*.

Vassiliades, A., Bassiliades, N., and Patkos, T. (2021). Commonsense reasoning with argumentation for cognitive robotics. In *RuleML+ RR (Supplement)*.

Venkatesh, S. G., Upadrashta, R., and Amrutur, B. (2021). Translating natural language instructions to computer programs for robot manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1919–1926. IEEE.

Zafar, H., Napolitano, G., and Lehmann, J. (2018). Formal query generation for question answering over knowledge bases. In *European semantic web conference*, pages 714–728. Springer.