Behavior-Tree Embeddings for Robot Task-Level Knowledge

Yue Cao and C.S. George Lee

***Abstract*— Recently, the behavior tree is gaining popularity as a robotic task-level knowledge representation. Manual design of behavior trees from scratch is tedious and cumbersome. Motivated by the need for an efficient way to reuse or transfer robot task-level knowledge, we propose a vector-space embedding approach that encodes a symbolic task into a numerical form. This approach, called behavior-tree embedding, takes a behavior tree that produces a single task as input and generates a corresponding vector. By exploiting the pre-trained language-embedding model and the node-aggregation mechanism, the produced embedding is capable of preserving both semantic information of task description and structural information of the hierarchical task organization. We evaluated the effectiveness and versatility of our proposed vector-space embedding approach in three different tasks.**

I. INTRODUCTION

Programming robots to perform complex tasks is a time-consuming task for robotic engineers/scientists. In an industrial deployment, it is common to re-program an entire task from scratch when new products are introduced [1]. According to the report on industrial robots from McKinsey & Company [2], robot programming accounts for about 25% of the total market of robot service. Since robot programming is laborious and costly, it is crucial to improve the efficiency of robot programming. In addition, the lack of homogeneous programming interfaces is ranked as the second major challenge for customers [2]. This suggests a potential cost saving can be realized by improving task-level programming efficiency in reusing or transferring task-level knowledge from previous usages.

Task-level programming plays an important role in robot programming. It coordinates complex abstract tasks into a sequential and/or hierarchical structure. In the past, various frameworks have been proposed for robot task-level programming, including state machines [3], Petri Nets [4], and SysML [5]. In recent years, Behavior Trees (BTs) have been gaining attention from the robotics community for task-level programming [6]–[8].

Behavior trees were initially developed by the video-gaming industry to coordinate the behavior of non-player characters in games. It has since achieved much success and become a common framework adopted by many game designers. In the past decade, behavior trees have also been extended and utilized as a tool for task representation in many robotic applications, such as CoSTAR [9] for manipulation tasks and Navigation2 [10] for navigation tasks. The modularity, reusability, flexibility, reactivity and expressiveness of behavior trees make them an alternative solution to the finite-state machine (FSM) and other frameworks. Different from the FSM that uses dense transitions among states, behavior trees represent transitions in a tree structure. Each subtree of a behavior tree is also a behavior tree, which makes it convenient to add, remove or modify subtrees without considering extra state transitions. Such modularity feature of behavior trees allows end-users to reuse or transfer task-level knowledge of robotic tasks.

To improve the task-level programming efficiency, one of the major challenges is to utilize or reuse vast task-level knowledge stored in a knowledge base and represented in behavior trees. This involves recognizing similar tasks and transferring them to new tasks. We propose to use the *vector-space embedding* technique for behavior trees as an efficient solution to model the task-level knowledge of robotic tasks for ease of re-usage and transfer of knowledge to new tasks with similar operations, as shown in Fig. 1.

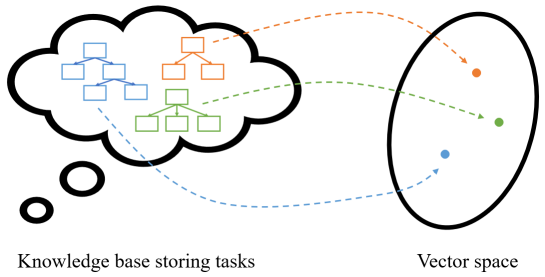


Fig. 1: Vector-space embedding for robot task-level knowledge. Given a knowledge base storing a collection of robot tasks, we seek to encode each task to a unique embedding in the vector space.

Vector-space embedding reduces a complex model to a low-dimensional vector while preserving its characteristics/features. Vectorial representation [11] was first proposed to overcome the limitations of the one-hot representation in document retrieval. In Natural Language Processing (NLP), vector-space models have been playing an important role since the last decade. The terminology "word embedding" was proposed to associate every word in the vocabulary with a real-valued vector [12]. Since the Word2Vec [13] technique has been developed to enable unsupervised learning on a large corpus of texts, the research on word embeddings in NLP has been growing dramatically. Since then the idea of vector-space embedding has been extended to other research areas to handle various types of entities, such as sen-

Yue Cao and C.S. George Lee are with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, U.S.A. Email: {yuecao, csglee}@purdue.edu.

†This work was supported in part by the National Science Foundation under Grant IIS-1813935. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

tences [14], graphs [15], and even trajectories of robots [16], [17]. These entities are difficult for machine-learning algorithms to cope with because they are not numerical. Hence, it is necessary to encode them into a vector space such that they can be integrated with machine-learning algorithms.

Inspired by the embedding technique in these areas, we propose a behavior-tree embedding approach that aims to encode robot tasks represented by behavior trees into compact vectors such that similar tasks can be placed closely in the vector space while distinct tasks stay far from each other. With our proposed behavior-tree embedding approach, the task-level knowledge of robot tasks can be reused or transferred in a more efficient way. In order to achieve this objective, we propose a 2-stage word-embedding-based approach with two major contributions:

(1) Both semantic and structural characteristics of behavior-tree-based robot tasks are preserved in the vector space, allowing more efficient usage and transfer of robot task-level knowledge.
(2) The behavior-tree embedding enables machine learning algorithms to be applied to symbolic task representation.

II. BACKGROUND

A. Behavior-Tree Fundamentals

Formally, a behavior tree is defined as a directed rooted tree $T = (N, E)$, with a node set N and an edge set E [8]. A behavior tree consists of a *Root*, branch nodes called control-flow nodes, and leaf nodes called execution nodes. In general, there are four types of control-flow nodes (*Fallback* ?, *Sequence* $\rightarrow$, *Parallel* $\Rightarrow$, and *Decorator* $\diamond$) and two types of execution nodes (*Action* and *Condition*). Here we list four commonly used node types with their characteristics [18] in Table I. For *Parallel* and *Decorator* nodes, we refer their details to [8], [18]. Note that all node names are written in italics for distinction.

TABLE I: Behavior-tree node types

Node	Success	Failure
Fallback	If one child succeeds	If all children fail
Sequence	If all children succeed	If one child fails
Action	When completed	Fails to complete
Condition	If true	If false

To execute a behavior tree, activation signals called "ticks" are generated from the *Root* node in a certain frequency and propagated through the whole tree. Once a leaf node is ticked, it executes the *Action* or verifies the *Condition* and returns its status – **Success, Failure** or **Running** – to its parent node. Eventually, the final status will be propagated all the way back to the *Root* node.

B. GloVe and USE Embeddings

Two specific language-embedding models are useful in the proposed behavior-tree embedding approach. Word embedding maps words into a low-dimensional vector space. GloVe [19] is a widely-used model for obtaining pre-trained word embeddings. It exploits the count-based matrix factorization and the context-based skip-gram model together. Global word-word co-occurrence matrix is added into the unsupervised training in order to capture global statistics of corpus. Several pre-trained GloVe embeddings are publicly accessible on `https://nlp.stanford.edu/projects/glove/`. They are trained on enormous corpora, such as 2 billion tweets from Twitter. With word-embedding techniques, words that have similar semantics are mapped closely together in the vector space.

Universal sentence encoder (USE) [20] is a general-purpose, sentence-embedding model that processes sentences to fixed-sized vectors. Two architectures are available for USE and we use the Transformer-based architecture only, which applies several transformer layers [21] in the encoding. Since the positional-encoding technique is applied during the training of the transformer, the ordering of words in one sentence is considered, which makes the output embeddings context-aware.

III. PROPOSED APPROACH

In this paper, we propose a behavior-tree embedding approach that aims to encode robot tasks represented by behavior trees into compact vectors such that similar tasks can be placed closely in the vector space while distinct tasks stay far from each other.

The proposed two-stage word-embedding approach models the task-level knowledge of robot tasks for ease of reusage and transfer of knowledge to new tasks with similar task operations. We present a baseline approach based on the sentence embeddings.

A. Problem Setting

We focus on the behavior tree that produces a single task. In other words, behavior trees structured by *Sequence* nodes and *Action* nodes will be taken into consideration, while behavior trees containing other node types including *Condition*, *Fallback*, *Parallel*, and *Decorator* nodes are excluded.

The principle of vector-space embeddings ensures that similar entities are nearby while distinct entities are far away in the vector space. Thereby, it is important to define the similarity for a certain entity. Considering the behavior-tree formalism, two characteristics are used to define the similarity for vector-space embeddings – structural and semantic characteristics. For the structural characteristic, we aim to preserve the feature of hierarchical organization of behavior trees in the vector space since it is associated with the modularity of behavior trees. For the semantic characteristic, we aim to keep the tasks that have similar linguistic semantics to be close in the vector space. In behavior trees, the textual descriptions of *Action* nodes contain linguistic semantic information and specify the primitive tasks to be executed. An example is shown in Fig. 2 to explain our problem setting.

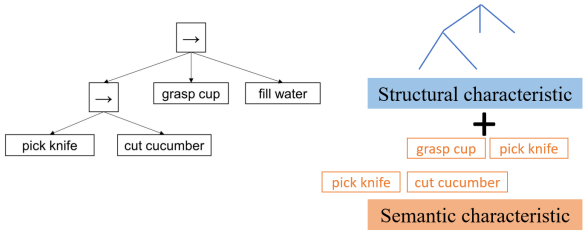


Fig. 2: Given a behavior tree consisting of *Sequence* nodes and *Action* nodes, our objective is to encode it into a vector while preserving both structural and semantic characteristics.

B. Word-Embedding-Based Approach

Our proposed approach for behavior-tree embedding is based on word embeddings and consists of two stages – word-embedding stage and node-aggregation stage.

Stage 1: Word-embedding stage

In this stage, we utilize the word-embedding technique to capture the rich semantic information of *Action* nodes. The pre-trained word-embedding model is applied to obtain the word vector for each constituent word in the *Action* node. Then we simply average the word embeddings of all constituent words in an *Action* node and obtain the node embedding,

$$v_n = \frac{1}{|n|} \sum_{w \in n} Enc(w) \tag{1}$$

where $w \in n$ is the constituent word of the *Action* node n, $|n|$ is the number of words in n, Enc is the word-embedding model, and v_n is the final node embedding of node n. For example, the node embedding for a "grasp ball" *Action* node is calculated by $(Enc(\text{grasp}) + Enc(\text{ball}))/2$. Averaging word vectors of text has demonstrated its effectiveness in representing phrases [22] and sentences [14] before.

Stage 2: Node-aggregation stage

Once the *Action*-node embeddings are computed, we carry out the node aggregation over the behavior tree. The node aggregation is conducted from the bottom to the top of the tree,

$$h_n^l \leftarrow AGG(\{h_u^{l+1}, \forall u \in Ch(n)\}) \tag{2}$$

where AGG is a node-aggregation function, h_n^l denotes the n^{th} node embedding in the l^{th} layer, $Ch(n)$ returns a set of indexes of child nodes. This process sums up the embeddings layer by layer. The final embedding at the root node serves as the embedding of the entire behavior tree.

The idea of node aggregation over the tree originates from [23], where an aggregation operation is performed in the field of medical ontology. This mechanism is similar to the message passing in a graph neural network [24], in which each node aggregates embeddings of its neighbors to update its new embedding. It allows a node to capture the structural information within its k-hop neighbors after the k^{th} message passing. Our node-aggregation operations propagate all structural information of the behavior tree to the root. The semantics of all primitive tasks are also concentrated into the final embedding vector. The aggregation function AGG can be either learnable or pre-fixed.

Given a behavior tree shown in Fig. 3, the overall algorithm is outlined in Algorithm 1 and illustrated in Fig. 4. In Stage 1, the text describing 4 primitive tasks is fed into a word-embedding model separately and turns into 4 vectors, v_1, v_2, v_3, and v_4. The acquired 4 vectors are used to assign

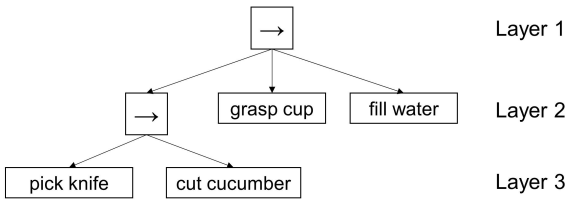


Fig. 3: A behavior tree with 3 layers.

values of the corresponding nodes in tree. Based on the tree structure, the embeddings of the nodes in the lowest layer, h_1^3 and h_2^3, are first sent to the aggregation function. The aggregated result h_1^2 is used as the embedding for their parent *Sequence* node. Afterwards, all 3 embeddings in layer 2 are passed through the aggregation function and output a final embedding h_1^1 for the root node.

Algorithm 1 Word-embedding-based approach

Input: A behavior tree T with L layers, pre-trained word embeddings Enc, a node-aggregation function AGG
Output: A behavior-tree embedding h_T

```
Stage 1:
for each Action node n in T do
    v_n ← (1/|n|) Σ_{w∈n} Enc(w)
end for
Stage 2:
Denote the embedding of the i^th node in the j^th layer in T as h_i^j
Initialize all Action node embeddings using v_n in Stage 1
for layer l = L − 1, L − 2, ··· , 1 do
    for n^th node in layer l do
        h_n^l ← AGG({h_u^{l+1}, ∀u ∈ Ch(n)})
    end for
end for
h_T ← h_1^1
```

The compositionality of our embeddings is well-suited for the modularity of a behavior tree. For example, consider we have two complete behavior trees T_1 and T_2, along with their embeddings h_{T_1} and h_{T_2}, respectively. When merging these two behavior trees into a new one, the embedding for the new behavior tree T_3 can be simply calculated by the aggregation function, that is, $h_{T_3} = AGG(h_{T_1}, h_{T_2})$.

C. Sentence-Embedding-Based Approach

We also present a sentence-embedding-based approach as a baseline measurement for the semantics of behavior trees. Given a behavior tree, we use tick signals to execute

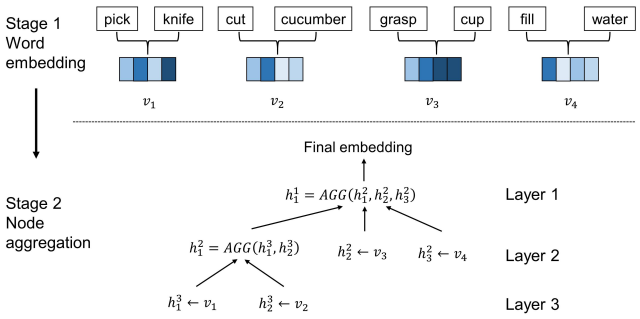


Fig. 4: Procedures for the word-embedding-based approach.

the behavior tree. Once the tree traversal is completed, an ordered sequence of *Action* nodes is acquired. Then we combine the text in the order sequence of *Action* nodes into one single sentence s. A sentence-level encoder Enc is applied to generate a sentence embedding $Enc(s)$. The process is described in Algorithm 2 and an example is illustrated in Fig. 5. This 3-layer behavior tree consisting of 4 *Action* nodes is "flattened" to one sentence with 4 phrases and 8 words – "pick knife", "cut cucumber", "grasp cup", and "fill water." This method has incorporated no tree structure information and will only be used to compare with the semantics of the word-embedding-based approach.

Algorithm 2 Sentence-embedding-based approach

Input: A behavior tree T, pre-trained sentence embedding Enc

Output: A behavior-tree embedding h_T

Run the behavior tree T

Store all executed *Action* nodes $n_1, \ldots, n_N$ in a traversal order

for each Action node n_i in T **do**

Obtain its phrase-description from its constituent words w: $p_i \leftarrow \{w_1, w_2, ...\}$

end for

Combine the phrase-description of sequenced *Action* nodes into a single sentence: $s \leftarrow \{p_1, p_2, ..., p_N\}$

$h_T \leftarrow Enc(s)$

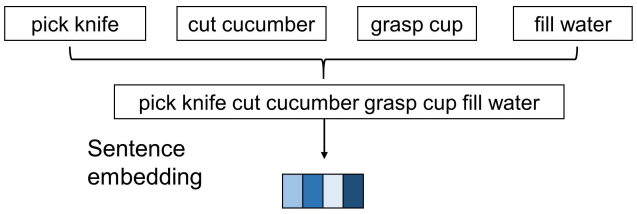


Fig. 5: Procedures for the sentence-embedding-based approach.

IV. EXPERIMENTAL EVALUATIONS & RESULTS

In order to assess the performance of our behavior-tree embedding method, we conducted three types of evaluation. First, we carried out a similarity measure to distinguish pairs of behavior trees semantically and structurally. Next, we applied the behavior-tree embeddings to a relatedness prediction task to showcase the bridge between symbolic tasks and downstream machine learning applications. Finally, we presented a case study on robot task knowledge transfer using the behavior-tree embeddings to reveal the connection with cognitive process.

A. Experimental Setting

We chose the pre-trained GloVe-twitter-200 as our word embeddings in the experiment. The GloVe-twitter-200 is a GloVe model trained on a corpus of 2 billion tweets. Each word in the GloVe-twitter-200 is represented as a 200-dimensional vector. We selected the pre-trained Universal sentence encoder based on Transformer architecture as our sentence-embedding model. It takes a sentence as input and outputs a 512-dimensional embedding. The embedding from the Universal sentence encoder is normalized in the output.

Due to the limit of training data in this work, we adopted a pre-designed node-aggregation function instead of a learnable function. Our aggregation function is defined as:

$$AGG(\{h_i, h_{i+1}, \cdots, h_j\}) = \frac{1}{j-i+1} \sum_{n=i}^{j} p(n-i) h_n, \quad (3)$$

where $\{h_i, h_{i+1}, \cdots, h_j\}$ is a set of child-node embeddings. These embeddings are sorted by their position from left to right. We also introduced a decay function $p(i) = e^{-\frac{i}{5}}$. The decay function is added to distinguish the cases, where child nodes are arranged in different orders. This differs from the permutation-invariant principle when defining message-passing functions in graph neural networks.

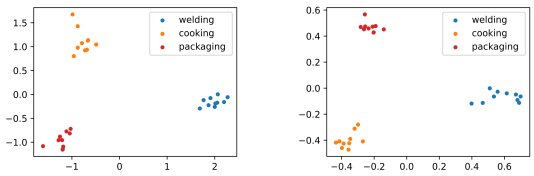
We also formalized the taxonomy for primitive tasks of robots. The *Action* nodes in the behavior tree represent primitive tasks and they contain rich semantics. We investigated 3 categories of generic robot tasks – welding, cooking, and packaging tasks. We used standard for welding process [25] to describe welding primitive tasks. We followed the atomic action taxonomy in [26] and designed the primitive cooking tasks. We utilized the task description from an industry report [27] and created the primitive tasks for packaging. Some of our synthesized primitive tasks are listed in Table II. Note that the task specification is not necessarily identical to our nomenclature. End-users have the flexibility to define their own set of primitive tasks as long as they share similar semantics. For instance, end-users can use the verb "chop" instead of "cut" to describe the same action. End-users can also alter primitive task "pick apple" to "grasp orange" since these two phrases stay close in the semantics space. We also generated 10 behavior-tree structures and placed the primitive tasks into them. We produced 30 behavior trees for these robot tasks in total, 10 for each task category.

TABLE II: A list of sample primitive tasks

Welding task	Cooking task	Packaging task
butt weld	pick knife	pick package
fillet weld	stir container	attach label
plug weld	slice salmon	pack product
spot weld	fill water	wrap film
seam weld	grasp plate	place board
⋮	⋮	⋮

B. Similarity Measure

We first started with visualizing the behavior-tree embeddings using the principal component analysis (PCA) for dimensionality reduction. The 200-dimensional embeddings from the word-embeddings-based approach and 512-dimensional embeddings from the sentence-embeddings-based approach are each projected into a 2-dimensional space. As shown in Fig. 6, the 30 behavior trees are separated in 3 clusters using PCA. The embedding for each behavior tree is located in the correct cluster.



(a) (b)

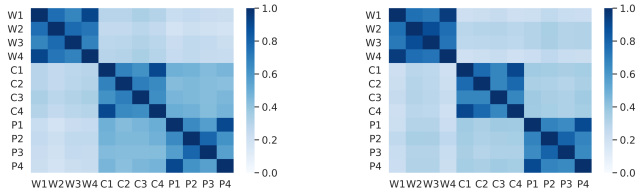
Fig. 6: PCA visualization of the behavior-tree embeddings of 30 tasks. We pre-designed 10 behavior trees for each robotic welding, cooking, and packaging task category. (a) The word-embedding-based approach. (b) The sentence-embedding-based approach.

Next, we studied the similarity between the behavior trees. Given the embeddings u, v of a pair of behavior trees, we then compute the angular cosine as the similarity measure between them:

$$\text{Sim}(u, v) = 1 - 2\arccos\left(\frac{u \cdot v}{\|u\|\|v\|}\right)/\pi. \tag{4}$$

The similarity measure ranges from 0 to 1, and the more similar pair gets a higher value. We illustrated some results of similarity computation in Fig. 7 as similarity matrices. We selected 4 behavior trees from each task category (i.e., welding, cooking and packaging tasks). We observed that the intra-category behavior trees have a high similarity value, mostly over 0.6, while similarity values of the majority of inter-category trees are below 0.4. The sentence-embedding-based method achieves better performance when comparing the cooking-task category and packaging-task category.

Because of the node-aggregation mechanism implemented in the process, the word-embeddings-based approach is capable of distinguishing behavior trees that are semantically similar but structurally different. The structural information of the behavior tree is mainly reflected by the magnitude of the embedding. For the sentence-based-embedding approach,



(a) (b)

Fig. 7: Similarity matrix for comparing different behavior trees, where W, C, and P stand for welding, cooking, and packaging tasks, respectively. (a) The word-embedding-based approach. (b) The sentence-embedding-based approach.

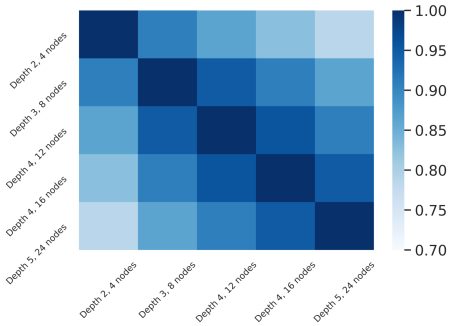


Fig. 8: Similarity matrix for comparison of behavior trees in different sizes.

each output of the Universal sentence encoder is normalized to 1, thus it does not include any information about sentence length. In other words, the sentence-embedding-based approach can only distinguish trees arranged in different orders (e.g., $T_1 \rightarrow T_2$ versus $T_2 \rightarrow T_1$), but not for trees in different sizes (e.g., $T_1 \rightarrow T_1$ versus T_1). We constructed a set of behavior trees that were semantically identical (e.g., using the same Action nodes) but in different structures. The similarity value calculated using Eq. (4) between them was over 0.99, thus we applied a new criterion to evaluate their structural similarity based on vector magnitudes:

$$\text{Sim}(u, v) = 1 - \frac{|\|u\| - \|v\||}{\max(\|u\|, \|v\|)}. \tag{5}$$

Figure 8 shows the similarity matrix computed by Eq. (5). Behavior trees with similar size share a high similarity value. By focusing on the magnitude of the embedding, the word-embedding-based method distinguishes behavior tress in different structures.

The results of similarity measure suggest that our proposed behavior-tree embeddings are capable of persevering the semantic and structural information of the behavior trees.

C. Relatedness Prediction

We also showed how the behavior-tree embeddings can be utilized in some downstream machine-learning tasks. We considered the relatedness prediction for pairs of behavior trees, which is a common supervised-learning task used

in NLP [28], [29]. Given a pair of behavior trees with embeddings u and v, we would like to predict a real-valued similarity score. In general, this similarity score is rated by a human judge. Since our main goal here is to show the adaptiveness of our embeddings rather than pursuing prediction accuracy, we simply assigned labels to different pairs instead of asking for human annotation.

We took the embeddings of 30 behavior trees (from the word-embedding-based approach) and performed the pair-by-pair computation between them. Specifically, given a pair of behavior-tree embeddings u and v, we calculated the element-wise product $u \odot v$ and distance $||u - v||$ between them. We assigned 4 different labels to each pair based on their category as shown in Table III. Hence, we obtained a total of 435 instances and split them into the training and validation sets as 348 and 87, respectively. We created 30 new behavior trees whose *Action* nodes are different from previous ones and processed them using the same operations. We also generated 435 instances and used them as the testing set.

TABLE III: Dataset overview

Pairs	similarity score	number of instances
C-W	1	100
W-P	2	100
C-P	3	100
W-W, C-C, P-P	4	135

Notation: C-W stands for cooking-welding task pair and W-P stands for welding-packaging task pair, etc.

We used the neural-network architecture shown in Fig. 9 for training. The inputs of the neural network are $u \odot v$ and $||u - v||$, which are 400-dimensional vectors. We expected to predicate a similarity score ranged in $\{1, 2, 3, 4\}$ for each pair. We used a multi-layer perceptron with 2 hidden layers as the classifier. The numbers of neurons in the hidden layers 1 and 2 and the output layer are 100, 25, and 4, respectively. The training was carried out by using a cross-entropy loss as the objective function and the SGD optimizer. We reported that the test-set accuracy was 0.961.

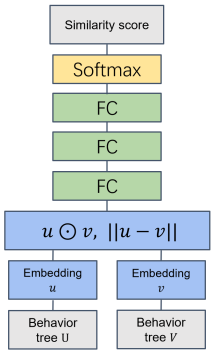


Fig. 9: Neural-network architecture for relatedness prediction of behavior trees.

In this experiment, our trained model took a pair of behavior-tree embeddings as input, and successfully generated a similarity score between them. The example of relatedness prediction exhibits the capability of behavior-tree embeddings to be applied in downstream machine-learning tasks.

D. Task Knowledge Transfer

We also carried out a case study to demonstrate that our proposed behavior-tree embeddings can support simple robot task knowledge transfer by vector arithmetic.

Consider a case of knowledge transfer of a behavior tree by modifying its sub-task. We built a source behavior tree shown in Fig. 10(a). We sought to transfer it to a target behavior tree shown in Fig. 10(b). Because of the modularity of behavior trees, the subtree replacement can be easily achieved without accounting for any transition change. This transfer operation can be considered as a knowledge transfer process in human analogy-making. In the example studied in cognitive science [30], children have the capability to transfer knowledge from "stack tires and stand on them" to "stack bales of hay and stand on them." We expected to simulate such human-like knowledge transfer, where our behavior-tree embedding serves as an abstract knowledge structure in the schema-based transfer process [31].

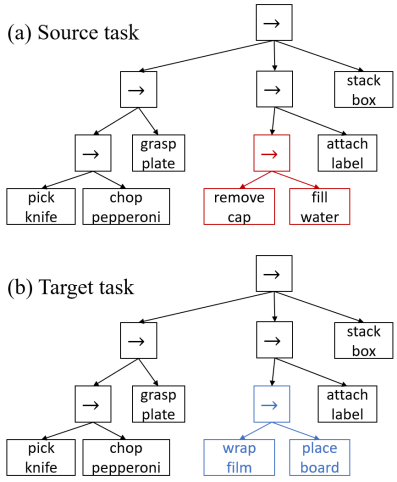


Fig. 10: A case study of task knowledge transfer. The red portion in the source behavior tree (a) is replaced by the blue portion in the target behavior tree (b).

We denoted the embeddings of the source and the target behavior trees as h_{src} and h_{tgt}, respectively. We assumed the embeddings of the red and blue subtree are h_{src_sub} and h_{tgt_sub}, respectively. The task knowledge transfer process can be represented by $h_{tgt} \approx h_{src} - h_{src\_sub}/6 + h_{tgt\_sub}/6$ in the vector space. For the given subtree, there are 1 sibling node in layer 3 and 2 in layer 1, which means that the subtree embedding is averaged by 2 firstly, then averaged by 3. We examined this relation by comparing the vector h_{tgt} with the vector $h_{src} - h_{src\_sub}/6 + h_{tgt\_sub}/6$. After the similarity computation using Eq. (4), we obtained $\text{Sim}(h_{tgt}, h_{src} - h_{src\_sub}/6 + h_{tgt\_sub}/6) = 0.975$ while $\text{Sim}(h_{tgt}, h_{src}) = 0.887$. Hence, we argued that the relation $h_{tgt} \approx h_{src} - h_{src\_sub}/6 + h_{tgt\_sub}/6$ holds.

This source-target embedding relationship indicates that symbolic task knowledge transfer can be possibly achieved in the vector space. Given a source task, a source sub-task and a target sub-task in the knowledge base, we can obtain their embeddings in the vector space and estimate the target task embedding using vector arithmetic. Once the approximate embedding of the target task is computed, we can perform similarity sorting and find the most relevant task in the knowledge base.

In this case study, we have verified that the task knowledge transfer can be represented by simple vector arithmetic. In fact, any manipulation including combination, removal, or modification of a behavior tree can be approximately represented by certain arithmetic expression of the embedding. Thus, it provides great flexibility for task knowledge transfer. Our case study also implies a connection between behavior-tree embeddings and knowledge abstraction in cognition.

V. SUMMARY AND CONCLUSIONS

This paper presented a novel vector-space-embedding approach for behavior trees. By leveraging the pre-trained GloVe word-embedding model and the node-aggregation mechanism, our proposed approach successfully mapped robotic tasks into a vector space while preserving their semantic and structural characteristics. This approach allows end-users to retrieve, reuse and transfer robot task-level knowledge. Moreover, the behavior-tree embedding bridges the gaps between the symbolic-task representation and machine learning.

Currently, our behavior-tree embedding approach is only limited to behavior trees consisting of *Sequence* and *Action* nodes. For further research, we plan to extend to behavior trees consisting of more node types. We will combine sentence-embedding-based approach with structural information of heterogeneous trees. Empirically, the context-aware sentence-embedding-based approach using the Universal sentence encoder outperforms the word-embedding-based approach in preserving semantic information. We plan to carry out a learning-based method to acquire an embedding for heterogeneous tree structure and concatenate it to the original sentence embedding.

REFERENCES

[1] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[2] M. Teulieres, J. Tilley, L. Bolz, P. Ludwig-Dehm, and S. Wägner, "Industrial robotics: Insights into the sector's future growth dynamics," *McKinsey & Company*, 2019.

[3] J. Bohren and S. Cousins, "The SMACH high-level executive [ROS News]," *IEEE Robotics Autom. Mag.*, vol. 17, no. 4, pp. 18–20, 2010.

[4] J. López, D. Pérez, and E. Zalama, "A framework for building mobile single and multi-robot applications," *Robotics and Autonomous Systems*, vol. 59, no. 3-4, pp. 151–162, 2011.

[5] J. Huckaby and H. Christensen, "Modeling robot assembly tasks in manufacturing using SysML," in *Proc. ISR/Robotik 2014; 41st International Symposium on Robotics*, 2014, pp. 1–7.

[6] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wąsowski, "Behavior trees in action: a study of robotics applications," in *Proc. ACM SIGPLAN Int. Conf. Software Language Engineering*, 2020, pp. 196–209.

[7] M. Colledanchise and L. Natale, "On the implementation of behavior trees in robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5929–5936, 2021.

[8] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and AI," *Robotics and Autonomous Systems*, vol. 154, p. 104096, 2022.

[9] K. R. Guerin, C. Lea, C. Paxton, and G. D. Hager, "A framework for end-user instruction of a robot assistant for manufacturing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 6167–6174.

[10] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst. (IROS)*, 2020, pp. 2718–2725.

[11] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[12] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[14] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[16] S. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," in *Proc. Conf. Robot Learning (CoRL)*, 2018, pp. 783–795.

[17] C. Devin, D. Geng, P. Abbeel, T. Darrell, and S. Levine, "Compositional plan vectors," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 14 963–14 974.

[18] D. Faconti and M. Colledanchise. (2018) BehaviorTree.CPP documentation. [Online]. Available: https://www.behaviortree.dev/

[19] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[20] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[22] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2013, pp. 926–934.

[23] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun, "GRAM: graph-based attention model for healthcare representation learning," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2017, pp. 787–795.

[24] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 1263–1272.

[25] *Welding and allied processes - Symbolic representation on drawings - Welded joints*, International Organization for Standardization Std. 2553:2019, 2019.

[26] M. J. Aein, E. E. Aksoy, and F. Wörgötter, "Library of actions: Implementing a generic robot execution framework by using manipulation action semantics," *Int. J. Robotics Research*, vol. 38, no. 8, pp. 910–934, 2019.

[27] ABB Robotics, "Robots for packaging industry, robot-based packaging automation," ABB Robotics, Tech. Rep., 2012.

[28] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[29] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. AAAI Conf. Artificial Intelligence*, 2016, pp. 2786–2792.

[30] A. L. Brown and M. J. Kane, "Preschool children can learn to transfer: Learning to learn and learning from example," *Cognitive Psychology*, vol. 20, no. 4, pp. 493–523, 1988.

[31] M. L. Gick and K. J. Holyoak, "Schema induction and analogical transfer," *Cognitive Psychology*, vol. 15, no. 1, pp. 1–38, 1983.