The NEMO co-pilot

Stefania Costantini[1,2,*,†], Pierangelo Dell'Acqua[3,†], Giovanni De Gasperis[1,†], Francesco Gullo[1,†] and Andrea Rafanelli[4,1,†]

[1]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
[2]Gruppo Nazionale per il Calcolo Scientifico - INdAM, Roma, Italy
[3]Department of Science and Technology, Linköping University, Linköping, Sweden
[4]Department of Computer Science, University of Pisa, Italy

Abstract
In this work, we describe an agent to be employed in Human-AI Teaming in various, even critical, domains, based upon affective computing, empathy, and Theory of Mind, and a description of the user profile and of the operational, professional, and ethical requirements of the domain in which the agent operates. The architecture of the proposed agent encompasses a Knowledge Graph, a Neural component and a Behaviour Tree. We briefly discuss a case study.

Keywords
Human-AI Interaction, Human-AI Teaming, Trustworthy AI, Responsible AI

1. Introduction

One recent focus in Artificial Intelligence (AI) is building intelligent systems where humans and AI systems form teams. This with the aim of exploiting the potentially synergistic relationships between human and automation, thus devising "hybrid" systems where the partners should cooperate to perform complex tasks, possibly involving a high degree of risk. As a simple example, in an AI-supported self-driving or assisted-driving vehicle, the AI component can be expected to evaluate and co-manage situations and risks, where the driver can provide the AI component with useful information on practical driving in all conditions and can self-manage the risks in the case this should be required by the circumstances. Human-automation interaction is, in fact, one of the main themes of Human-centered AI. This issue also falls in the realm of Trustworthy AI, whose requirements include respect for human autonomy, prevention of harm, fairness, and explainability, and of Responsible AI, whose goal is to employ AI in a safe, trustworthy and ethical fashion.

AI and humans, if working together in Human-AI Teaming (HAIT), can produce results exceeding what either can achieve alone, whereas they can control and improve each other. For instance, a human driver might train to cope with previously unseen situations through co-driving automation via a cooperative task shared between the human driver and the AI-based system installed on the vehicle. At the same time, AI helps drivers in case of difficulties and immediate risks. In this synergistic relationship, humans may improve automation efficacy and capabilities. At the same time, automation may enhance human performance in a task and compensate for human inadequacies, catching and correcting possible misbehaviors, possibly also due to physically or emotionally impaired states, and providing valuable suggestions.

For the tasks of adopting AI agents in crucial tasks such as, e.g., improving caregiving in medicine and teaching and constructing effective human-AI teams, agents should be endowed with an emotion recognition and management module, capable of empathy, and modelling aspects of the Theory of Mind (ToM), in the sense of being able to reconstruct what someone is thinking or feeling. Modelling a Theory of Mind is often based on forms of "Affective Computing", which is a set of techniques aimed at eliciting a human's emotional condition from physical signs, to enable the system to respond intelligently to human emotional feedback.

In this work, we describe an agent to be employed in HAIT, based upon affective computing, empathy, and Theory of Mind, and a description of the user profile and of the operational, professional and ethical requirements of the domain in which the agent operates. The architecture of the proposed agent encompasses a Knowledge Graph, a Neural component and a Behavior Tree.

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy
*Corresponding author.
†These authors contributed equally.
stefania.costantini@univaq.it (S. Costantini); pierangelo.dellacqua@liu.se (P. Dell'Acqua); giovanni.degasperis@univaq.it (G. D. Gasperis); francesco.gullo@univaq.it (F. Gullo); andrea.rafanelli@phd.unipi.it (A. Rafanelli)
http://www.di.univaq.it/stefcost (S. Costantini); https://dellacqua.se/ (P. Dell'Acqua); https://fgullo.github.io/ (F. Gullo)
0000-0002-5686-6124 (S. Costantini); 0000-0003-3780-0389 (P. Dell'Acqua); 0000-0001-9521-471 (G. D. Gasperis); 0000-0002-7052-1114 (F. Gullo); 0000-0001-8626-2121 (A. Rafanelli)

2. Background

2.1. Behavior Trees

Behaviour Trees (BTs) were introduced as a tool to enable modular AI in computer games. A behavior tree is essentially a mathematical model of plan execution, where each element (task and action) of a plan is associated with a node in the tree. Their strength comes from their ability to create complex tasks composed of simple tasks without worrying about how the simple functions are implemented. For a comprehensive survey of BTs in Artificial Intelligence and Robotic applications, see [1, 2]. A BT is a directed acyclic graph consisting of different types of nodes, each one associated with executable code (where such code enacts an element composing a plan). In most cases, a BT is tree-shaped, hence the name. However, unlike a traditional tree, a node in a BT can have multiple parents, allowing the reuse of that part of the tree. The traversal of a behavior tree starts at the top node. When a node is traversed, the associated code is executed, returning one of the three states: *success*, *failure*, or *running*.

The critical nodes in a BT include leaf nodes and inner nodes. An *action* is a leaf node representing a behavior that the character can perform. The action returns success or failure when it completes its execution, depending on the outcome. An action is depicted as a white circle. A *condition* is a leaf node that checks an internal or external state. It returns either success or failure. A condition is represented as a grey rounded rectangle. A *sequence selector* is an inner node that typically has several child nodes that are executed sequentially. Once a child node completes its execution successfully, the sequence selector continues executing the next child node. If every child node returns success, then the sequence selector returns success. If one of the child nodes return failure, the sequence selector immediately returns failure. A sequence selector is depicted as a grey square with an arrow across the links to its child nodes. A *priority selector* is an inner node. It has a list of child nodes that it tries to execute one at a time until one of the child nodes returns success. If none of the child nodes executes successfully, the priority selector returns failure. A priority selector is represented with a grey circle with a question mark.

2.2. Neural Empathy-Aware Behavior Trees

To consider empathy and mimic human decision-making, in [3] we introduced *neural empathy-aware behavior trees* (NEABTs) by introducing a selector node called *emotional selector*, an *empathy node*, and a *neural node*.

The emotional selector is a node that orders its child nodes based on the agent's current affective state. The agent elaborates on the affective state during repeated interactions with the user and then tune its reaction accordingly. Once the ordering has been established, the emotional selector behaves as a priority selector. A white circle with the character E represents an emotional selector. In contrast, an empathy node provides an emotional evaluation of its single child node. An empathy node can only be a child of an emotional selector. Its child can be a leaf or an inner node. A dashed circle line with the name of the empathy emotion represents an empathy node.

To enable the integration of deep learning models for emotion recognition and symbolic models for planning and decision-making within emotional behavior trees, we introduced *neural nodes*. A neural node takes the current state of the environment and agent as input and, using a deep learning model, makes inferences about the emotional state. It contains a model, such as an emotion recognition system, that estimates the emotional state. These estimates are then mapped into the agent's affective state variables that parameterize the emotional selector. The neural node continually updates the agent's internal emotional state, allowing the dynamic adaptation of behavior trees to the emotional context.

2.3. Knowledge Graphs

Knowledge Graphs (KGs) are sets of *facts* (i.e., *triples* such as "Dante," "wrote," "Divine Comedy") that interconnect *entities* ("Dante," "Divine Comedy") via *relationships* ("wrote") [4, 5]. Entities and relationships correspond to nodes and (labelled) edges of the KG, respectively. KGs have been extensively used in a plethora of application scenarios, including knowledge completion [6], head/-tail prediction [7], rule mining [8], query answering [9], and entity alignment [10, 11, 12]. KGs are also known as knowledge bases [13], information graphs [14], or heterogeneous information networks [15].

A noteworthy technique that is commonly exploited for tasks on KGs is *Knowledge graph embeddings* (KGEs) [16], which aims at generating a vector representation for entities and relationships of a KG.

3. Framework

The architecture of the proposed agent is illustrated in Figure 1. The main components of the architecture are the *User*, the *Environment*, a *Knowledge Graph* (KG), and a *Behavior Tree* (BT). The overall interaction between such components is described next.

The BT is fed with signals from Environment, User, and KG. Such signals are exploited by the BT to perform its computation and to output (*i*) an action to be suggested to the User, (*ii*) an action actually performed by the agent

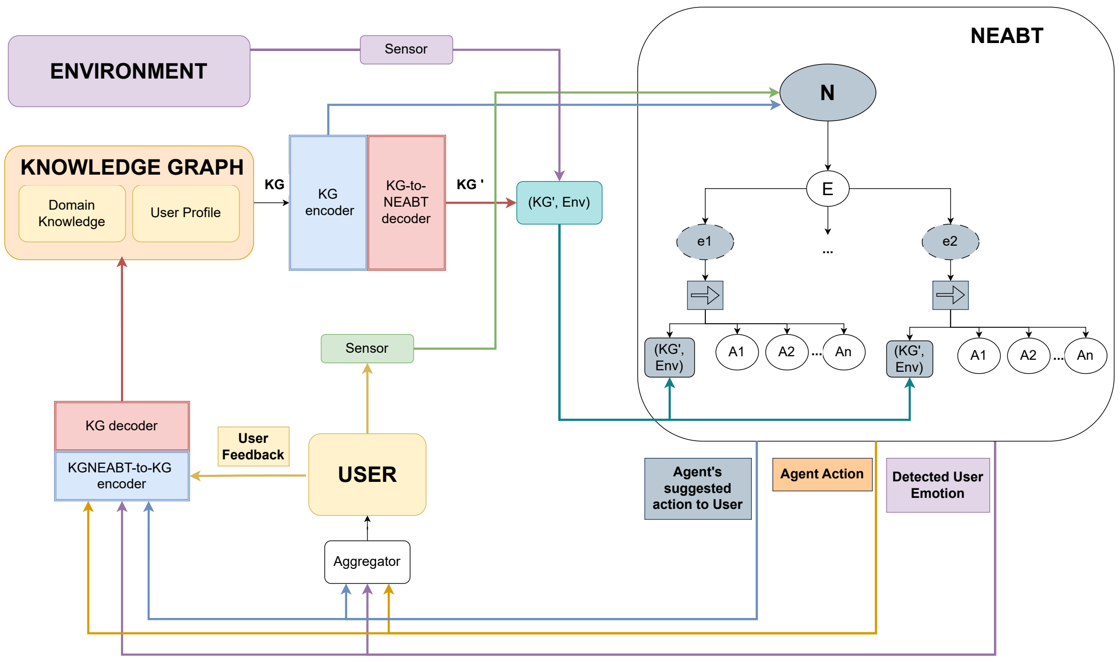

Figure 1: The NEMO co-pilot framework

(e.g., an empathetic action), and (*iii*) User's emotion detected by its neural node ('N', see below). Threefold BT's output passes through an "Aggregator", responsible for suitably aggregating and presenting three BT's outputs to the User. The Aggregator may perform something either very simple (e.g., just derive a textual representation of the three outputs and concatenate them) or more sophisticated (e.g., exploit a large language model (LLM)). BT's outputs and User's feedback are used to update back the KG. This way, we have a loop-back mechanism in which the KG is exploited by the BT for its internals, and the BT is exploited to update the KG properly.

Next, we describe the User, Environment, KG, and NEABT components in more detail.

User. The User performs reactions and actions based on the signals provided by the NEABT. The user's sensory data flow into the NEABT through a sensor, which represents them in some proper numerical format. Also, the User's feedback—e.g., whether (or to which extent) the User has adopted the Agent's suggested action—is sent back to the KG. User's reactions/actions are assumed to be determined by all three types of BT's output. In particular, the User's emotion detected by the BT at the previous iteration is important for establishing the emotional conditions that most influence the user.

Environment. Signals from the surrounding environment are detected by a sensor, representing them in some numerical format, and are thus ready to be processed by the NEABT (along with the KG representation).

KG. The KG contains information about domain knowledge and user profile. KG's information is provided to the NEABT in a twofold form. It is first encoded in some proper numerical format and passed to BT's neural node (see below). The encoding is performed by a KG encoder component, which can be implemented, e.g., with a KGE (see Section 2.3). KG's encoded information is then decoded into a format suitable for processing by the internal nodes of the BT. A KG-to-BT decoder performs KG's information decoding. This can be implemented, e.g., as a neural network component whose training can be performed on a ground truth defined through either manual annotation or the agent's historical data. The KG is fed BT's output and user feedback. Such data in input to the KG are represented in a format suitable for updating the KG, e.g., a set of KG triples should be added and a set of KG triples removed. Such a translation from BT's and User's signals to KG updating signal is performed by a further encoder-decoder component. Again, such an encoder-decoder can be implemented as a neural network and trained with a ground truth defined manually or through historical data.

NEABT. The NEMO framework deploys a NEABT as a behavior tree. The BT's neural node receives the KG's information and the user's sensory data and makes inferences about the user's emotional state. These estimates are mapped into the user affective state variables that parametrize the neural node child, the emotional selector. In turn, the emotional state selector passes the values of the affective state variables to its child nodes, empathy nodes. Each child empathy node provides an empathic evaluation of its subtree. In Figure 1, every subtree has a root node that is a sequence selector with a condition node as a child and several action nodes. The condition child node returns success/failure by performing a test condition upon the input pair (KG', Env). The corresponding action child nodes are executed if the condition node returns success. By doing so, the NEABT can execute actions over the environment. Some of these action nodes define the BT threefold output.

4. Case Study: Driver Co-Pilot

Here, we envision a case study that involves developing an intelligent agent that actively functions as a "companion" (co-driver) and support system for drivers. The agent will assist drivers by providing interventions in risky situations that may arise due to external circumstances and/or the driver's health condition and emotional state, taking into account emotional aspects that could impact driving performance.

The intelligent agent will also be trained through interaction with the human user following the recent "Human-AI teaming" paradigm. A human driver could cooperatively train the agent by collaboratively performing various driving-related tasks, even under challenging scenarios. In this synergistic relationship during the training phase, humans enhance the effectiveness of automation (capabilities and performance). At the same time, the agent installed in each vehicle improves human efficiency and compensates for human inadequacies by intercepting and correcting potential erroneous behaviours, possibly resulting from compromised physical or emotional states.

Potential intervention modes for the agent to assist a struggling driver could include automatically activating (semi-)autonomous driving mode (if available) so the driver can momentarily divert their attention. Alternatively, the agent could more actively engage with the driver to regain attentiveness, such as by recommending stimulating music on a dedicated radio station. In case of health issues, the agent could recommend pulling over to rest or take medication (e.g., for hypertension) or, in critical cases, seek emergency assistance by contacting emergency services.

References

[1] M. Colledanchise, P. Ögren, Behavior trees in robotics and AI: An introduction, CRC Press, 2018.

[2] M. Iovino, E. Scukins, J. Styrud, P. Ögren, C. Smith, A survey of behavior trees in robotics and ai, Robotics and Autonomous Systems 154 (2022) 104096.

[3] S. Costantini, P. Dell'Acqua, G. De Gasperis, A. Rafanelli, Empowering emotional behavior trees with neural computation for digital forensic, 15th European Symposium on Computational Intelligence and Mathematics (ESCIM 2024) (in press).

[4] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, ACM CSUR 54 (2022) 71:1–71:37.

[5] G. Weikum, Knowledge graphs 2021: A data odyssey, PVLDB 14 (2021) 3233–3238.

[6] X. Wang, L. Chen, T. Ban, M. Usman, Y. Guan, S. Liu, T. Wu, H. Chen, Knowledge graph quality control: A survey, Fundamental Research 1 (2021) 607–626.

[7] S. Ji, S. Pan, E. Cambria, P. Marttinen, S. Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, Trans. Neural Netw. Learn. Syst. 33 (2021) 494–514.

[8] B. Yang, S. W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: ICLR, 2015.

[9] Y. Wu, Y. Xu, X. Lin, W. Zhang, A holistic approach for answering logical queries on knowledge graphs, in: ICDE, 2023, pp. 2345–2357.

[10] S. S. Bhowmick, E. C. Dragut, W. Meng, Globally aware contextual embeddings for named entity recognition in social media streams, in: ICDE, 2023, pp. 1544–1557.

[11] J. Huang, Z. Sun, Q. Chen, X. Xu, W. Ren, W. Hu, Deep active alignment of knowledge graph entities and schemata, PACMMOD 1 (2023) 159:1–159:26.

[12] A. Zeakis, G. Papadakis, D. Skoutas, M. Koubarakis, Pre-trained embeddings for entity resolution: An experimental analysis, PVLDB 16 (2023) 2225–2238.

[13] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, A. Doan, Building, maintaining, and using knowledge bases: a report from the trenches, in: SIGMOD, 2013, pp. 1209–1220.

[14] M. Lissandrini, D. Mottin, T. Palpanas, D. Papadimitriou, Y. Velegrakis, Unleashing the power of information graphs, ACM SIGMOD Record 43 (2015) 21–26.

[15] C. Shi, Y. Li, J. Zhang, Y. Sun, S. Y. Philip, A survey of heterogeneous information network analysis, TKDE 29 (2016) 17–37.

[16] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, TKDE 29 (2017) 2724–2743.