Open Research Online

The Open University's repository of research publications and other research outputs

Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain

Journal Item

How to cite:

Meloni, Antonello; Angioni, Simone; Salatino, Angelo; Osborne, Francesco; Reforgiato Recupero, Diego and Motta, Enrico (2023). Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain. IEEE Access, 11 pp. 22468–22489.

For guidance on citations see FAQs.

© [not recorded]

https://creativecommons.org/licenses/by/4.0/

Version: Version of Record

Link(s) to article on publisher's website:
http://dx.doi.org/doi:10.1109/access.2023.3253388

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data policy on reuse of materials please consult the policies page.

oro.open.ac.uk

IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 30 January 2023, accepted 26 February 2023, date of publication 6 March 2023, date of current version 9 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3253388

RESEARCH ARTICLE

Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain

ANTONELLO MELONI[1], SIMONE ANGIONI[1], ANGELO SALATINO[2], FRANCESCO OSBORNE[2,3], DIEGO REFORGIATO RECUPERO[1], AND ENRICO MOTTA[2]

[1]Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy
[2]Knowledge Media Institute, The Open University, MK7 6AA Milton Keynes, U.K.
[3]Department of Business and Law, University of Milano-Bicocca, 20126 Milan, Italy

Corresponding author: Diego Reforgiato Recupero (diego.reforgiato@unica.it)

ABSTRACT In the last few years, chatbots have become mainstream solutions adopted in a variety of domains for automatizing communication at scale. In the same period, knowledge graphs have attracted significant attention from business and academia as robust and scalable representations of information. In the scientific and academic research domain, they are increasingly used to illustrate the relevant actors (e.g., researchers, institutions), documents (e.g., articles, patents), entities (e.g., concepts, innovations), and other related information. Following the same direction, this paper describes how to integrate conversational agents with knowledge graphs focused on the scholarly domain, a.k.a. Scientific Knowledge Graphs. On top of the proposed architecture, we developed AIDA-Bot, a simple chatbot that leverages a large-scale knowledge graph of scholarly data. AIDA-Bot can answer natural language questions about scientific articles, research concepts, researchers, institutions, and research venues. We have developed four prototypes of AIDA-Bot on Alexa products, web browsers, Telegram clients, and humanoid robots. We performed a user study evaluation with 15 domain experts showing a high level of interest and engagement with the proposed agent.

INDEX TERMS Chatbots, knowledge graphs, human–robot interaction, scholarly data, user experience, virtual assistant.

I. INTRODUCTION

In the last few years, chatbots have become mainstream solutions adopted in a variety of domains for automatizing communication at scale. They have become an important tool for supporting users in answering their questions and performing a number of tasks among which customer care [1], [2], ordering items [3], booking tickets [4], giving driving directions [5], and so on. Generally, they adopt techniques such as natural language understanding and generation to interpret the user's question, building an equivalent query to a knowledge base and returning information to the users according to the results of the generated query [6].

The recent evolution of artificial intelligence (i.e., neural networks) has led to the development of more advanced conversational agents which provide more effective dialogues, flexibility, and interactions with the users[1] [7]. Besides, with the widespread usage of smart sensors connected to the Internet and apps, conversational agents acquired new sources where to take contextual information and operate. This has pushed users to use them for daily tasks such as turning on home devices, managing calendars, etc.

Examples of advanced AI-driven virtual assistants with the aforementioned features are Apple Siri,[2] Google Now,[3] Microsoft Cortana,[4] Amazon Alexa,[5] Wit.ai,[6] and Snips.ai.[7]

The associate editor coordinating the review of this manuscript and approving it for publication was Mehedi Masud.

[1]https://apo.org.au/sites/default/files/resource-files/2016-09/apo-nid210721.pdf
[2]https://www.apple.com/siri/
[3]https://assistant.google.com/
[4]https://www.microsoft.com/en-in/windows/cortana
[5]https://developer.amazon.com/alexa
[6]https://wit.ai/
[7]https://snips.ai/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Conversational agents are being employed in many different domains. For example, in the e-learning field, they provide novel and smart communicative capabilities, hence improving teaching activities: they were able to give motivation and engagement to the students so that they could increase their learning by gaining meta-cognitive skills [8].

Within public administrations, conversational agents have been employed for different tasks. For example, a recent Italian platform for job postings offers a chatbot providing job recommendations based on the users' skills [9]. Another work shows a chatbot framework designed to answer questions related to services offered by a public administration [10]. In such a case, some of the challenges were related to the big set of services proposed by the public administration, their intricacy, the domain (public administration), how the users formulated their questions, and the differences between the language used by technical persons (e.g., lawyers, bureaucrats) and by lay people. Van Noordt et al. [11] gave an exploratory insight of three different chatbots employed within the public administrations of Vienna, Latvia, and Bonn. They assessed that the employment of chatbot technology in public administrations is correlated with minor organisational changes.

Within the health domain, there is great attention towards recent discoveries within AI technologies with the goal to automatize services in health centers such as nursing homes and hospitals [2]. The work described by Callejas and Griol [12] illustrates some of the applications of conversational interfaces concerning the mental health sphere. Furthermore, the survey published by Montenegro et al. [13] investigates approximately 4,145 articles related to conversational agents in health published from 2009 to 2019. Even within the aerospace domain, conversational agents have been proposed to offer quick and concise answers to complex situations. For example, Liu et al. [14] mixed a task-oriented dialogue system with a conversational agent and an interactive question-answering component: the objective was to assess the benefit of smart search and conversational search for cockpit documentation.

Recently, we have witnessed an increasing diffusion and employment of Knowledge Graphs (KGs), which are becoming a standard solution for representing complex interconnected data. KGs acquire and integrate information from the real world by using an ontology. In particular, it represents the information by means of a graph whose nodes are entities whereas edges represent their relationships [15]. This formal and structured representation allows automatic programs to better interpret users' questions.

In this context, the new challenge is to design chatbot architectures able to access and operate together with KGs and extend the range of queries that users are allowed to express in order to identify and return the information they might be interested in. For example, Bockhorst et al. [16] present an approach to developing task-oriented conversational interfaces that construct a system of grammar to correctly infer parses from natural language. The system grammar is built by leveraging the structured types and entities of an underlying KG complemented by a machine learning-driven restructuring procedure. Developing a new generation of chatbots able to capitalize on knowledge graphs is thus the natural but challenging step forward.

KGs are increasingly used also for representing the scholarly domain. These KGs typically represent the relevant actors (such as institutions and researchers), scientific reports (such as patents, and research articles), entities (such as concepts, technologies, and tasks to solve), as well as additional information (such as funded projects, acknowledgments) in a structured organization. Nowadays, in the literature, we can find many large-scale KGs within the scholarly domain such as OpenCitations [17], Scopus,[8] Semantic Scholar,[9] Aminer [18], CORE,[10] ORKG,[11] OpenAlex,[12] and others. Given the amount of data and analytics that can be inferred and analyzed out of such a large amount of information represented by scholarly data, conversational agents could be leveraged to interact with the users and provide the information that is being analyzed. In fact, a smart conversational agent would be extremely useful in this domain, both for supporting the day-to-day job of scientific researchers and for allowing other users to obtain factual information about the scientific landscape and recent research outcomes. However, while there is an abundance of literature about conversational agents in education [19], to the best of our knowledge, works on conversational agents for supporting academic and scientific research are not present within the scientific literature.

This means that, although conversational agents have proved to be very effective for interacting with users (especially with the latest ChatGPT language models[13]), their responses in specific domains, such as scientific research, often includes misleading or not factual statements. For instance, if we ask the well-known ChatGPT to *"List the top ten papers about blockchain sorted by citations"*, it will return a mix of both real and completely fictional articles, in which several titles and/or authors do not appear in any repository. The numbers of citations also appear to be incorrect. The fact that these systems can confidently suggest fake scientific information is very problematic. In other cases, the system will at least admit its ignorance and refuse to answer. For instance, if we ask ChatGPT the question "*Count papers which were written by computer scientists from Princeton University*", it will return the following text, which is technically correct, although very generic, but does not provide the answer expected by the user:

> *It is not possible for me to accurately count the number of papers written by authors from*

[8]Scopus - https://www.scopus.com/
[9]Semantic Scholar - https://www.semanticscholar.org/
[10]https://core.ac.uk/
[11]https://www.orkg.org/orkg/
[12]OpenAlex - http://openalex.org
[13]https://openai.com/blog/chatgpt/

Princeton University, as I do not have access to current information or browsing capabilities. Additionally, it would be difficult to determine which papers were written by authors from Princeton University, as there may be many papers with multiple authors from different institutions or with no institutional affiliation listed. It would also be challenging to determine which papers to include in the count, as there is a vast amount of research published in various fields and it is not clear what specific criteria should be used to determine which papers should be included.

We thus have to equip existing conversational agents with "knowledge plugins" so that they can produce accurate and verifiable answers in specific domains.

In order to address this crucial challenge, this paper presents a novel approach for integrating conversational agents with knowledge graphs and produces factual and relevant answers in a specific domain. On top of the proposed architecture, we developed AIDA-Bot, a chatbot focused on the scholarly domain. AIDA-Bot capitalizes on a large-scale knowledge graph of scholarly data for answering natural language questions on research papers, concepts, authors, institutions, and scientific conferences. Specifically, it utilizes the Academia/Industry DynAmics (AIDA) Knowledge Graph[14] [20], which contains over 21 million scientific papers annotated using the research topics of the Computer Science Ontology (CSO)[15] [21]. For example, AIDA-Bot is able to answer the two queries discussed above by respectively producing 1) an accurate list of the top ten articles regarding blockchain and 2) the number of articles written by computer scientists from Princeton University (e.g., *"I found 14,806 papers from authors affiliated to Princeton University."*).

To prove the versatility of the proposed architecture, we developed four prototypes of AIDA-Bot. One of them runs on Alexa devices and is implemented via the Amazon Alexa Skill Kit. The second one is a lightweight web application that can be run on browsers and uses widespread web technologies (JQuery, Javascript, Web SpeechAPI). The third chatbot is implemented in Python and runs on Telegram. The fourth one uses the Choregraphe suite and relies on a NAO humanoid robot that can interact verbally and physically with the user for more advanced human-robot interaction. We discuss these implementations and share the codebase in order to allow researchers and developers to easily reuse our architecture.

Finally, we report the results we have obtained from a user study involving 15 domain experts. The user study had two main purposes. First, we wanted to examine for the first time the interaction between researchers and a conversational agent able to respond to questions within the scholarly domain and assess the perceived utility of this solution. Second, we intended to evaluate the specific implementation of AIDA-Bot to derive useful feedback for future developments.

This paper is built on top of a 4-page poster paper presented at the International Semantic Web Conference [22]. The differences between the proposed paper and the poster paper, which also correspond to the innovations we bring in this manuscript, are highlighted in the following:

- an extended version of the architecture (i.e., the chatbot engine and the corresponding data server module have been greatly improved);
- we define a number of questions that can be asked using different English grammar (and not just using one single form as in [22]) and answered using knowledge graphs within the scholarly domain;
- we describe in detail and share the codebase of four implementations of AIDA-Bot for browsers, Alexa devices, Telegram clients, and humanoid robots;
- we introduce one new query type: *compare*, allowing to list similarities and differences of two instances, such as researchers or organizations;
- a comprehensive literature review, which was not included in the poster paper;
- we assess the user experience and usability (not present in [22]) of AIDA-Bot through a user study involving 15 computer scientists.

The remainder of this manuscript is structured as follows. In Section II, we report previous related work on conversational agents and scholarly knowledge graphs. Section III introduces the architecture of the chatbot. Section IV describes and justifies the four kinds of queries (i.e., *count*, *list*, *describe*, and *compare*) that our conversational agent implements. Section V discusses the main interaction scenarios between the user and the chatbot. Section VI describes the four user entities we have developed. Section VII delineates the outcome of the qualitative evaluation involving 15 domain experts. Section VIII explains how to extend the conversational agent with new question types.

Finally, Section IX reports final remarks and future research directions where we are headed.

II. RELATED WORK

In the following, we will discuss the state of the art regarding the two main topics of this paper: conversational agents and scholarly knowledge graphs.

A. CONVERSATIONAL AGENTS

Chatbot technology has always been attractive to researchers, starting in 1966 when Joseph Weizen-Baum developed ELIZA.[16] Early technologies, such as ELIZA, used keyword matching and context identification to chat with users

[14] Academia/Industry DynAmics Knowledge Graph - http://w3id.org/aida/

[15] Computer Science Ontology - http://w3id.org/cso/

[16] https://en.wikipedia.org/wiki/ELIZA

but were unable to carry on long conversations. ALICE[17] (Artificial Linguistic Internet Computer Entity) is another well-known historical chatbot winning the Loebner Prize award on three occasions (2000, 2001, and 2004). This chatbot is based on the Artificial Intelligence Mark-up Language (AIML),[18] which is a lightweight and highly configurable language and still supports many of today's chatbots [23].

Practitioners keep developing and studying new features to improve the functionalities of the existing methods, and at times they have introduced new architectures. These new developments leverage ontologies and context, i.e., details about both current and previous conversations [24].

Chatbots can be categorized according to a set of characteristics: i) knowledge domain, ii) type of interaction, iii) usage, and iv) design techniques [25]. The last describes the design philosophy behind a chatbot and how different categories of chatbots deal with the conversation in a given context.

When considering their objectives, chatbots may be categorized into two main classes: task-oriented [26] and non-task-oriented [7]. Task-oriented chatbots are tailored to specific scenarios like booking a hotel, flight, or accommodation, ordering goods, planning events, or helping users in accessing a given information [27]. They are designed to support users in achieving a specific goal in a circumscribed domain, but they lack general knowledge. On the other hand, non-task-oriented chatbots are mainly purposed for lengthy conversations, work in the open domain, and are built for imitating the characteristics of human-human unstructured conversation [28].

We can further distinguish chatbots depending on their interaction mode (text-based vs. voice-based). Text-based chatbots interact through text messages. They aim to quickly identify users' needs and provide them with instant solutions. They are typically used by businesses to handle the interaction with customers [29]. One advantage they have is their flexibility to be integrated with social media, messaging apps, and so on. Conversely, voice-based chatbots [30], [31] are able to recognize human speech and respond with synthetized speech. Some examples are personal assistants, such as Amazon Alexa, Google Assistant, Siri from Apple, and Cortana from Microsoft. These are often used for task-oriented applications, such as searching for information on the Internet, making calls, sending text messages, playing multimedia content, interacting with IoT devices, and telling jokes [32].

It is also possible to characterise chatbots according to their engine (rule-based vs AI-based). Rule-based chatbots [33] use a tree-like flow to help users with their questions. This means that they guide the user with follow-up questions to eventually get the correct response. The structures and answers are typically predefined. Other chatbots employ AI and natural language processing techniques [34], that, unlike rule-based chatbots, do not use keywords, patterns, or rules to determine the user's intent, but try to infer it directly from the text.

Sometimes, chatbots are tailored to work in specific domains such as i) healthcare [35], ii) education [36], and iii) business [37]. Chatbots in healthcare support patients and their relatives by answering specific health-related questions on HIV/AIDS [38], child health [39], and mental health [40], to name a few [35]. For example, Divya et al. [41] developed a medical chatbot for self-diagnosing diseases, which provides also detailed descriptions of them. Additional chatbots in healthcare include MedChatbot [42] and Mandy [43]. The former is used to support medical students. The latter is used by healthcare workers to automate patient intake. Other chatbots collect information about people's diet [44] or provide restaurants with a tool to collect allergy information based on users' allergens [45]. Chatbots in education support the teaching of a variety of subjects, such as English [46], Medicine [42], and business process models [47]. Some chatbots are also able to answer university-related questions that are typically found in FAQs [48]. The reader is referred to [36] for a review of works on the use of chatbots in education. Finally, in the business domain, there are chatbots supporting companies in their daily tasks [37]. For example, chatbots were developed to support customer service for businesses and e-commerce [1], [3], helping to complete certain tasks [49], and improve user experience [3]. Works presented in a recent workshop [50] discussed innovative techniques to interact with chatbots, understand conversations, promote mental health and well-being, improve the coverage of clarification responses, assess chatbot applications in different domains, and measure how a chatbot can be supportive or engaging.

In the last few years, we saw the emergence of a variety of conversational agents and question-answering systems that build on semantic web technologies and knowledge graphs [51], [52], [53], [54], [55]. The main advantage of these solutions is their ability to integrate and formulate complex queries on heterogeneous data from multiple sources [56], including large-scale general knowledge bases such as Wikidata [57] and DBpedia [52]. They also support reasoning and link prediction techniques [58] for identifying and correcting errors as well as enriching the knowledge base with new facts [59]. This allows a conversational agent to act accordingly to a flexible representation of information that can easily get updated by seamlessly including new data, entity types, and semantic relations [60], [61]. For this reason, many high-profile conversational agents take now advantage of large-scale knowledge graphs, such as the Google Knowledge Graph and the Alexa Knowledge Graph.

Although, several other domains have been affected by the introduction of chatbots, to the best of our knowledge, we still lack chatbots able to target the scholarly knowledge domain, and support the several stakeholders in this space,

[17] A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) - https://en.wikipedia.org/wiki/Artificial_Linguistic_Internet_Computer_Entity

[18] http://www.aiml.foundation/

such as researchers, students, research policymakers, and companies. For instance, these solutions could support users in analysing trends in the literature, choosing a venue in which to disseminate their work, finding possible collaborators, identifying relevant articles, and so on. The architecture and the prototype presented in this paper aim at addressing this gap.

B. SCHOLARLY KNOWLEDGE GRAPHS

We have recently observed the increasing development of many large-scale knowledge graphs representing research papers and their related metadata. Some examples are Microsoft Academic Graph (MAG) [62], the OpenAIRE research graph[19] [63], OpenAlex,[20] AMiner [64], Open Research Knowledge Graph (ORKG)[21] [65], the Artificial Intelligence Knowledge Graph (AI-KG)[22] [66], SciGraph,[23] ScholarlyData,[24] PID Graph[25] [67], and OpenCitations[26] [17].

MAG [62] is a heterogeneous scholarly knowledge dataset developed and maintained by Microsoft. It describes more than 548M entities including scientific publications, authors, topics, institutions, journals, and conferences, as well as all the relationships between them. It is the largest dataset of scholarly data freely available, and it includes over 260M publications [68]. Since January 2022, MAG has been replaced by OpenAlex. Dimensions is a dataset developed by Digital Science, that consists of 122M research publications from various disciplines, 140M patents, as well as data about grants, policy documents, and clinical trials [69]. The Academia/Industry DynAmics (AIDA) [70] is a knowledge graph built with the purpose, as the name suggests, of analyzing the dynamics between academia and industry. The current version integrates 25M research papers from MAG, and 8M patents from Dimensions, characterizing them according to the CSO topics [71], the types of affiliations (i.e. academic, industrial, or collaborative effort), and the industrial sectors of INDUSO.[27] Since Microsoft recently decommissioned MAG, AIDA is now switching to a combination of OpenAlex and DBLP, which offers comparable coverage.

Scopus is a scholarly dataset developed by Elsevier, consisting of more than 80M research papers. Public administrations and funding bodies consistently adopt it to calculate metrics of performance and impact. Although it is well-curated, according to Visser et al. [68] its paper coverage is not as comprehensive as MAG. The Semantic Scholar Open Research Corpus[28] [72] is a dataset of about 202M research papers powering the Semantic Scholar search engine and released by the Allen Institute for AI. DBLP[29] is a dataset of publications in Computer Science, which was initially developed by the University of Trier and now is managed by Schloss Dagstuhl. Currently, it includes metadata on 5.5M articles, 2.7M authors, 5.4K conferences, and 1.7K journals. The OpenCitations Corpus (OCC) [17] is a dataset of 69M bibliographic resources and 1.1B citation links. It is maintained by OpenCitations, an organization aiming at publishing bibliographic data and citations using semantic web technologies. The AMiner Graph [64] is a dataset with over 320M publications powering the AMiner system. AMiner is an academic search engine that mines the online profiles of researchers and structures them according to the metadata. The OpenAIRE dataset DOIboost[30] [73] is another integration effort enhancing Crossref with information from Unpaywall,[31] MAG, and ORCID[32] covering authors, organizations, and abstracts. The Open Academic Graph[33] is a scholarly dataset integrating MAG and AMiner. The present version (v.2.1) contains 183M papers from AMiner and 240M papers from MAG, with an intersection of 119M articles. Some knowledge graphs focus in particular on representing the content of research articles. For instance, the Artificial Intelligence Knowledge Graph (AI-KG) [66] offers 1.2M statements extracted from over 300K publications describing five types of entities (tasks, methods, metrics, materials, and others). The Open Research Knowledge Graph (ORKG) [65] includes a description of over 10K articles according to relevant topics, approaches, datasets, and evaluation methodologies.

We developed AIDA-Bot on top of the AIDA knowledge graph since it integrates several datasets (MAG, Dimensions, DBLP, CSO, and INDUSO) and offers a sophisticated representation of research areas [74]. It thus enables users to ask questions on a variety of entities, such as authors, topics, conferences, journals, countries, affiliations, and industrial sectors. However, as already mentioned, the current architecture is highly general, making it easy to switch to other knowledge graphs. It must however be remembered that the knowledge graph defines the kind of questions the bot can answer. For instance, since Semantic Scholar does not offer a good representation of conferences, a chatbot using this data source will not be able to answer questions on conferences.

III. THE CHATBOT ARCHITECTURE

In the following, we will introduce and discuss the general and flexible architecture for chatbots depicted in Figure 1. This solution takes as input the user's question, interprets it,

[19] OpenAIRE research graph - https://graph.openaire.eu
[20] OpenAlex - https://openalex.org/
[21] Open Research Knowledge Graph - https://www.orkg.org/orkg
[22] AI-KG - https://w3id.org/aikg
[23] SciGraph datasets - https://sn-scigraph.figshare.com
[24] ScholarlyData - http://www.scholarlydata.org
[25] PID Graph - https://www.project-freya.eu/en/pid-graph/the-pid-graph
[26] OpenCitations - https://opencitations.net
[27] Industrial Sector Ontology (INDUSO) - https://aida.kmi.open.ac.uk/downloads/induso
[28] ORC - http://s2-public-api-prod.us-west-2.elasticbeanstalk.com/corpus/
[29] DBLP - https://dblp.org
[30] DOIboost last release - https://zenodo.org/record/3559699
[31] Unpaywall - https://unpaywall.org
[32] ORCID - https://orcid.org
[33] Open Academic Graph - https://www.openacademic.ai/oag/

and, if needed, asks for further clarification, then it translates the question to a query that can be run on the knowledge base and finally produces an answer based on the results of the query. It is structured into three parts: i) User Entity, ii) Chatbot Module, and iii) Data Module.

The *User Entity* is the module that controls the communication with the user either by voice or text. It is typically either an Instant Messaging technology (e.g., Telegram and Facebook Messenger), a browser (e.g., Google Chrome), a smart speaker (e.g., Amazon Echo), or a robot (e.g., NAO, Pepper).

The *Chatbot Module* takes care of clarifying the question by asking for missing information and generating the answer. It is composed of two main components: i) the Chatbot Engine and ii) the Language Logic module. The Chatbot Engine coordinates the communications between the User Entity and the Data Server and verifies that the user query is well-formed and contains all the information needed.

In Figure 2 we can see that the Chatbot Engine is structured according to a model based on a finite state automaton. At the beginning of each new session, the bot is in state 0 where it needs to check what intent (query) is contained in the user input. If the identified intent allows an immediate response (i.e., "hello", "reset", or "help"), then the bot directly performs the relative task. If the intent is more complex (i.e., count, list, describe, or compare), and, therefore, an instance has to be identified, the system passes to state 1 and then to one among C ("count"), L ("list"), D ("describe"), or E ("compare"), depending on the identified intent. Since states D, E, and L act similarly to state C, they are not detailed in Figure 2. Focusing on the C ("count") intent:

1) If one or more necessary input parameters are missing, the system asks them to the user in state C.0.
2) In state C.1, the system verifies the existence of an instance (i.e., an author, a topic, a conference) in the knowledge base, its correct classification, and the presence of homonyms.
3) If multiple candidates have been identified for an input instance (e.g., the topic 'Neural Networks' vs the topic 'Feedforward Neural Network' when looking for the token 'neural'), the bot manages the case in state C.3 by asking a clarification question to the user.
4) If a case of homonymy has been identified (e.g., two authors with the same name), the system manages the case in state C.4 by asking for further information from the user.
5) In state C.2, the bot produces the response and then returns to state 0 for a new session.

The Language Logic processes the output of the Data Server for a given query and produces a well-formed answer in natural language. In particular, the Language Logic module takes care of analyzing the user input and assigning it to one of the query types through a decision tree, based on the presence of some key tokens (e.g., "count", "enumerate", "how many") or their most common synonyms. The key tokens are then stripped from the input and the remaining groups of words are searched in the knowledge graph to identify possible instances (author, topic, organization, etc.). Therefore, the module takes the user input and returns the type of query (if possible) and the potential instances and their types found in the knowledge graph. The Language Logic module also takes care of building the response in natural language starting from the data obtained from the knowledge graph. To do this, we created a dictionary containing, for each type of query, all the possible combinations of data types. For each combination, it returns the most suitable verbs, prepositions, and articles for constructing the answer in natural language.

Finally, the Data Module interprets the user's question and provides an interface to query the knowledge base. Within the Data Module we have the Web Server with a Parser and a Query dictionary, and the Data Server with the Knowledge Base. The Web Server handles the communication between all components. The Parser analyses the user statement identifying the elements needed to recognize the type of the query so that it can be formally generated using the Query dictionary. In addition, the Parser communicates with the Knowledge Base to check the entities retrieved from the text (e.g., *artificial intelligence*, *Massachusetts Institute of Technology*) and identifies their types (e.g., *topic*, *journal*).

In order to exemplify how this architecture works in a realistic scenario, we will use AIDA-Bot, the prototype we created that implements this architecture in the domain of scholarly data. AIDA-Bot relies on the Academic/Industry DynAmics Knowledge Graph (AIDA). AIDA incorporates information from DBpedia, the Computer Science Ontology, Microsoft Academic Graph, the Global Research Identifier Database, and Dimensions. It is freely available under CC-BY 4.0 as a dump and can be queried using SPARQL from the http://w3id.org/aida/sparql triplestore. AIDA describes the metadata about 21M publications in Computer Science. All the articles are also classified with the research concepts drawn from CSO. By leveraging AIDA, users can ask questions about six classes of entities (*authors, topics, papers, conferences, citations*, and *institutions*) and their instances (e.g., *University of Maryland, ESWC, Machine Learning*).

At the beginning of each session the chatbot welcomes the user and asks if they have any question.

The user can then ask a question through the *User Entity*. Voice-enabled interfaces will need to incorporate a speech-to-text service to convert voice messages into text. Next, the *User Entity* sends the question to the Chatbot Engine within the *Chatbot Module*, which in turn sends the message to the *Parser* to extract all the parameters from the question.

There are three kinds of parameters: *instance*, *class*, and *query-building parameters*. An instance is an entity described in the knowledge graph, such as "Machine Learning" or "University of Oxford". In presence of ambiguity (e.g., *ISWC* is an acronym adopted by different conferences), the chatbot will prompt back to the user by asking to select one of

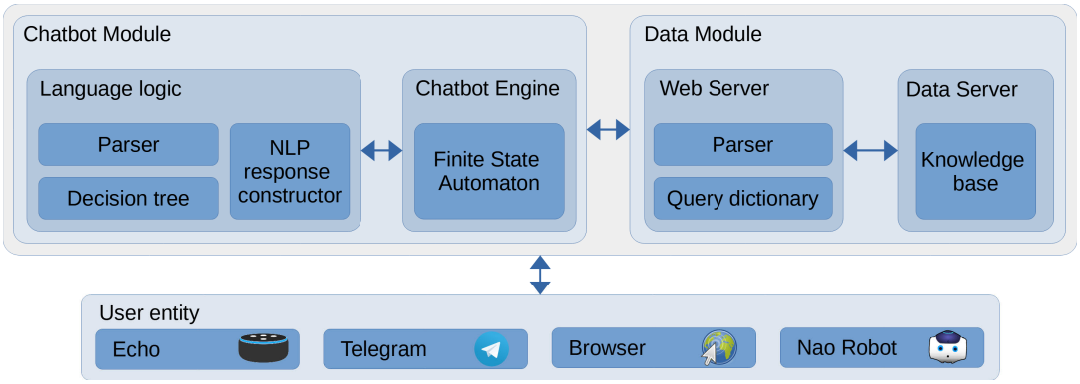


FIGURE 1. System architecture.

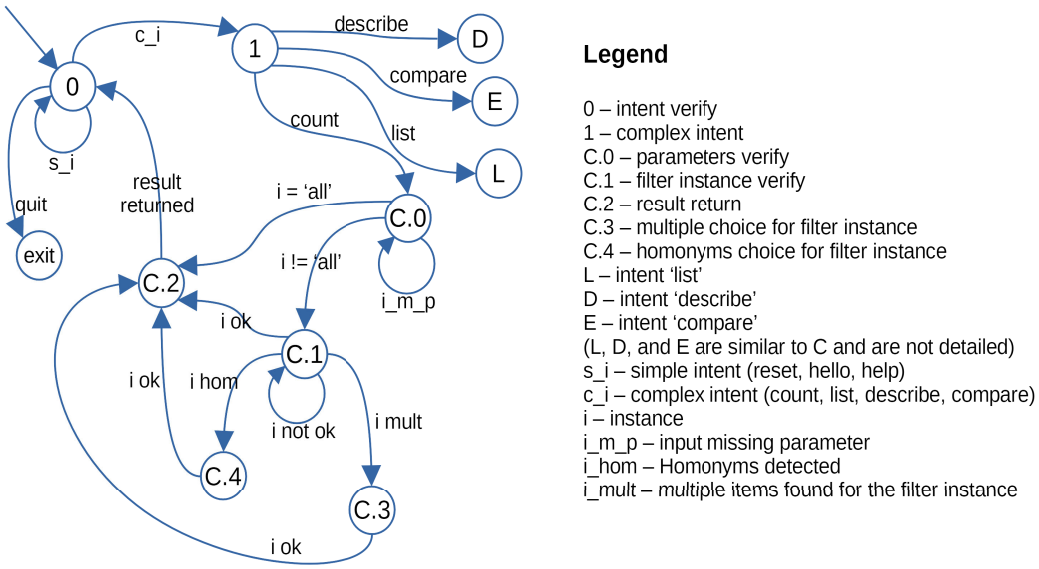


FIGURE 2. Chatbot engine - finite state automaton.

the possibilities (e.g., *International Symposium on Wearable Computers, International Semantic Web Conference*). The class refers to the type of this instance, such as "Topic" or "Organization". The system will try to automatically recognize each instance (e.g., Machine Learning) and assign them to the relevant class in the knowledge graph (e.g., Topic). If the system is unable to find a perfect match, it identifies the entity with the lowest Levenshtein distance from the input token, in order to correct potential spelling mistakes. For example, searching for "machine learnin" will return the topic "machine learning". The query-building parameters are the value that allows the chatbot to understand the specific user query. We consider three kinds of query-building parameters: *keywords*, which specify the kind of query (e.g., "list", "describe"), *classes of the queried items*, which identify the type of items to be returned by the query (e.g., "Organizations", "Papers"), and *orders*, which define the order of the returned entities (e.g., "by the number of citations"). The last one applies only to queries that return a list of items. Table 1 reports the query-building parameters used by AIDA-Bot.

As an example, the user's request ≪count the number of papers from the University of Cagliari≫ will be parsed as in the following: "count" and "papers" are the query-building parameters (keyword and class of queried items, respectively), "University of Cagliari" is the instance, and "Organization" (implicitly detected by the system) is the class of the instance "University of Cagliari". The chatbot will use these parameters to formulate a formal query to the knowledge graph for counting all the items of class "Paper" associated with the entity "University of Cagliari", which is an "Organization".

Users may ask incomplete questions, e.g., ≪count the number of papers≫. If this is the case, the Chatbot Engine gets back to the user and asks for the missing details. For instance, in the previous example, the chatbot is missing the name of the instance that identifies the context of the *count* query. Therefore, it uses a template to generate a message that asks for this specific information, e.g., ≪What is the name of the topic, conference, organization, or author whose papers I should count? You can say 'all' for the full list.≫. The user can then answer with the required information,

TABLE 1. Query-building parameters for AIDA-Bot. The character / means that the parameter is not available for the given query.

query	keywords	classes of queried items	orders
count	count, how many	papers, authors, conferences, organizations, citations	/
list	list, enumerate	papers, authors, conferences, organizations, topics	publications, citations, publications in the last 5 years, citations in the last 5 years
describe	describe, who is, what about, what is, what, who	/	/
compare	compare, match	/	/

e.g., ≪Artificial Intelligence≫ if they want to request the number of articles in *Artificial Intelligence*. As soon as all the elements are available, the query is formed and sent from the Chatbot Engine to the Data Server. The latter executes the query on the Knowledge Graph and returns the output.

The Chatbot Engine employs the Language Logic module for producing a well-formed answer based on the query results and forwards it to the *User Entity*.

If the system fails to recognize the intent, the bot will engage the user with a guided procedure to identify the desired intent.

A. SEARCHING THE KNOWLEDGE BASE

In this section, we will give further details on how to correctly identify the desired information in the knowledge base. An exact search returns the best results, but users do not always remember exactly the entity name to search and thus can make some syntactical mistakes (typos, misspellings, etc.). By using a fuzzy search we tackle such potential problems.

Any interrogation of the knowledge base to identify an entity starts with an exact search. The search is successful if it returns at least one entity and its associated type (class). If no entities are found, the system tries to perform approximate matches by using the Levenshtein distance.

The similarity score is calculated based on the Levenstein distance and the maximum length of the source and the target texts. Let l be the maximum length of source and target texts and d the Levenstein distance. The score s will be: $s = (d - l)/l$.

As an example, let us assume that the user types "rdwf geaph" as input. The system creates a set of all possible search term variations within an experimentally determined edit distance. For the example above, it would create a list of 496 term variations using "2" as the maximum edit distance (e.g., "rdf geapq", "trdf geaph", "rdfy geaph", 'rdfg eaph", "rdf graph" and so on). Then, for computational purposes, they are sorted in decreasing order of similarity score with the aim to get the first 50. The Knowledge Base then returns exact matches for each one of them.

This fuzzy search may produce a significant number of results. The system uses a multilevel threshold with a value higher than the one initially chosen for the search, experimentally determined to limit the number without excluding potentially valid results. If the similarity value of a single result is higher than the threshold value, the system returns it as a validated result. If the process produces multiple outcomes, the system will return a maximum of three options per class and ask the user to select the correct one. In our example, the system would return the term "rdf graph", found in the knowledge base as a topic and having a similarity score of 0.8 with the input sentence. If the number of results with a similarity score above the threshold is too high, the system returns a warning for the user, inviting him to be more specific.

IV. THE USER QUERIES

The architecture presented in the previous section can support many kinds of user queries. In this section, we discuss in detail the four types of queries that we developed for AIDA-Bot: *count*, *list*, *describe*, and *compare*.

The engine automatically identifies several syntactic forms of the queries above. The user may even start a sentence with a query keyword such as *count* to enable the conversational agent that will interactively ask all the other needed parameters.

A. COUNT QUERY

The *count* query allows users to assess the number of elements of a class C within a context identified by an instance I. As an example, ≪count papers on natural language generation≫ or ≪count authors in NeurIPS≫. The results can be sorted in 4 different ways: by the number of publications, by the number of citations, by the number of publications in the last 5 years, and by the number of citations in the last 5 years.

Figure 3 illustrates an example of the count query. We can see that the system first welcomes the user who is invited to perform a query. The user then asks: ≪count the papers on machine learning≫. In such an example, C corresponds to "papers" and I corresponds to "machine learning" (an instance of the *Topic* item). The system automatically identifies *Topic* by searching through all the data and identifying one only occurrence of the term "machine learning". Once the user confirms the query, the system replies with the number of papers in the database associated with the *Topic* "machine learning".

Then the system prompts again to the user to ask another query. The user says ≪count the papers≫. This request is not complete and the system asks for the I term of the query which has not been provided. The user inputs "Giorgia" but this term is very general and is available (by performing an approximate match against the AIDA knowledge graph) as

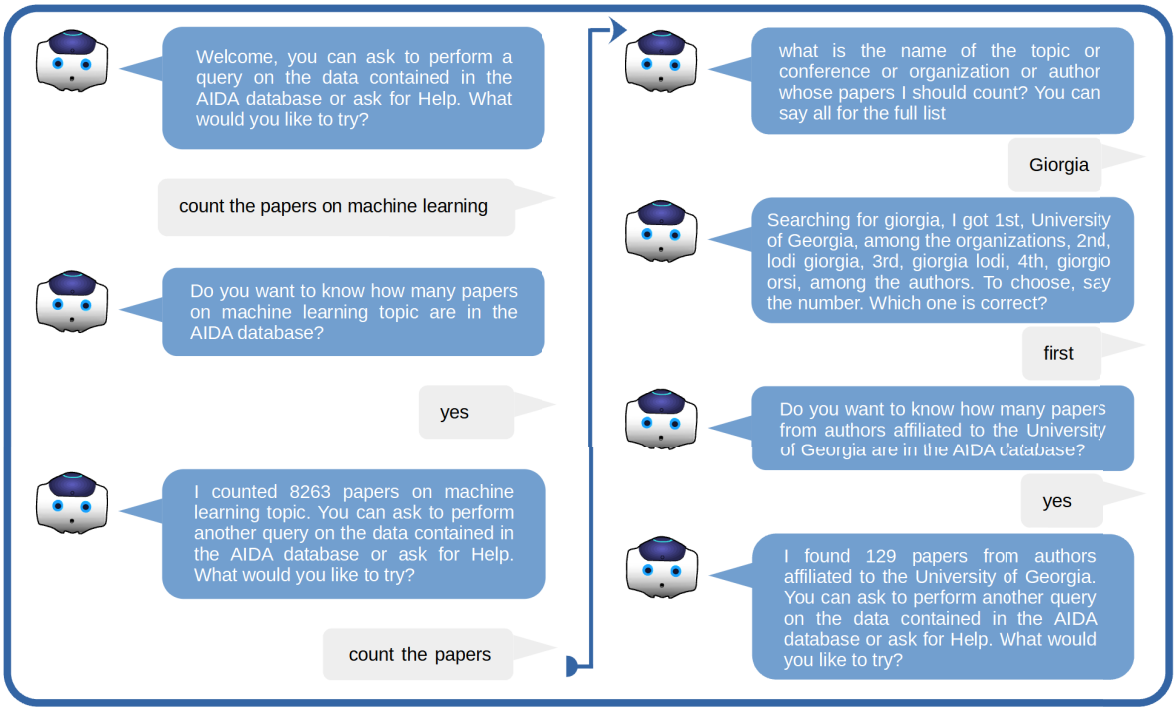


FIGURE 3. Example of interaction for the *count* query.

different items of type *I*. If the total number of potential candidates is less than ten, the system lists them all asking the user to choose among the listed ones. In our example, the term "Giorgia" is present four times: one is an institution and three are authors. The user specifies the first choice corresponding to an organization, and the system returns the results of the implicit query ≪count the papers of the University of Georgia≫. In case the total number of candidates is higher than ten or there are three different types *I*, the system asks the user to better specify their request instead of listing all the possible choices.

B. LIST QUERY

The *list* query allows users to retrieve the top *N* items of a certain class *C* according to a sorting *O* within a context specified by an instance *I*. As before, the results can be sorted in 4 different ways: by the number of publications, by the number of citations, by the number of publications in the last 5 years, and by the number of citations in the last 5 years. Some examples include ≪list the top 10 papers in Artificial Intelligence sorted by citations in the last five years≫ or ≪list the top 5 authors in NeurIPS ordered by citations≫.

Figure 4 depicts an example of the *list* query. The first message always welcomes the user. Then, the user asks simply ≪list 2≫, providing the number 2 for *N* without any other information and the system responds by asking the user to fill in the item of class *C*. Please note that *N* is not mandatory; if not provided, the system assumes 3 as the default value. Also, users can insert the question as a whole, but for ease of understanding, we are breaking it down into different steps. Next, the user chooses ≪papers≫, and the system then asks them to fill the element of class *I*. The user inputs a conference acronym: ≪iswc≫. The system automatically recognizes *iswc* as a conference, by searching through all the data in AIDA, but it finds two conferences with the same acronym. To this end, it needs to disambiguate such a case by performing the same operations described at the end of the paragraph related to the *count* query. In our example, as the number of retrieved elements is two and of the same class, the system lists them asking the user to choose one. Finally, the system asks the user to specify the order, by choosing one of the allowed sorting options. After the user confirms their query, the system replies with the list of the top 2 papers presented at the "International Semantic Web Conference" sorted by the number of citations in the last 5 years.

C. DESCRIBE QUERY

The *describe* query allows users to obtain the description of a certain instance *I* of class *C* (author, conference, or organization). It uses a different template for each class.

For example, ≪describe Yoshua Bengio≫ will produce a description of the researcher Yoshua Bengio based on a number of metrics (e.g., h5-index, citations) including his areas of expertise and the conferences where he published the most. Similarly, ≪describe NeurIPS≫ will return a description of the conference according to relevant bibliometrics and the lists of the most involved universities,

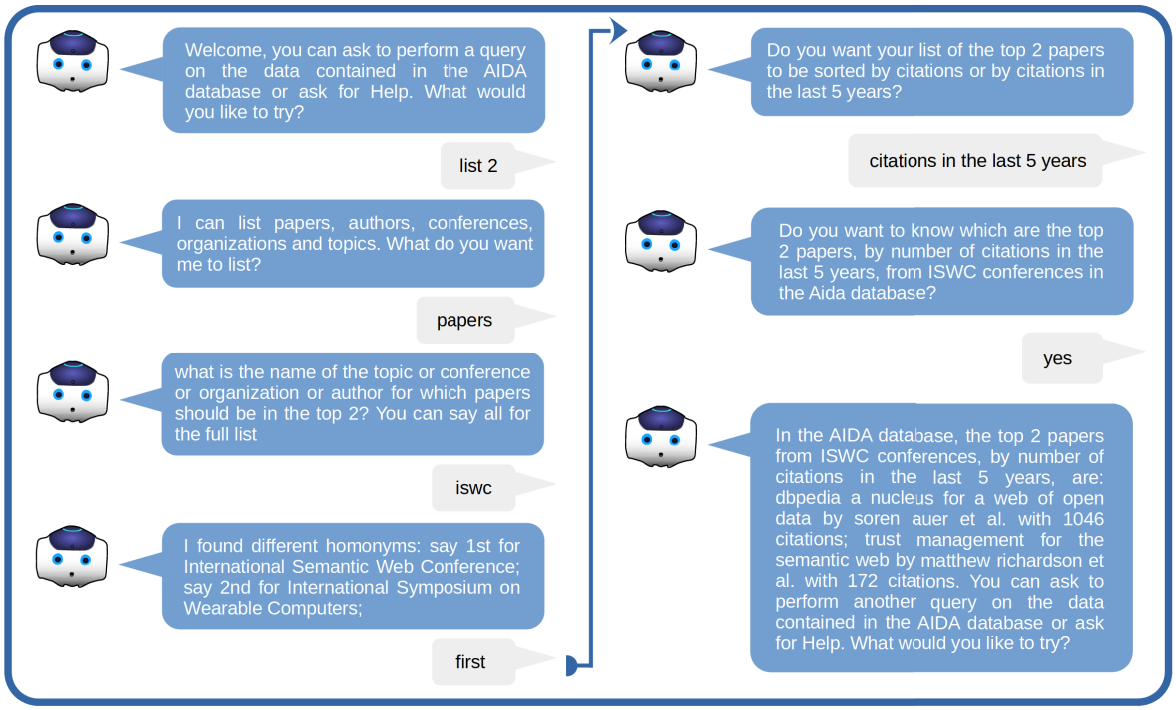


FIGURE 4. Example of interaction for the *list* query.

companies, and countries. The describe query is currently available only for organizations, conferences, and authors, but we are working on expanding it to the other classes.

Figure 5 illustrates an example of the *describe* query. After the usual welcoming message of the system, the user asks ≪describe EKAW≫. The system recognizes the acronym of the conference and requests confirmation from the user. After the user's confirmation, the system replies with the information about the conference. Then the system prompts again to the user to ask another query. The user says ≪describe≫ and the system asks for the name of an author or a conference. The user writes ≪Bizer≫ and the system finds Christian Bizer as the only author present in the database with the requested last name and asks for confirmation from the user. After the user's acknowledgment, the system returns the information of the describe query for the requested author.

D. COMPARE QUERY

The *compare* query allows users to ask for a comparison of two instances I_1 and I_2 of class C (author, conference, or organization). For the sake of consistency, it is only possible to compare instances of the same type; if not so, the user is prompted by the chatbot.

When comparing two researchers, the chatbot returns a comparison of their number of publications, number of citations, h-index, as well as whether they share research topics. It also comments on which author performs better in each category. Conferences are instead compared according to their starting years, number of citations in the last five years, h-index, and the main research topics. Finally, organizations are compared according to their number of publications and citations in the last five years, h-index, and the main topics.

Figure 6 illustrates an example of the *compare* query. After the welcoming message, the user asks ≪compare ESWC to EKAW≫, which are the acronyms of two major conferences in the field of Semantic Web. The system recognizes the two conferences from the acronyms and asks for confirmation. Upon confirmation, the system replies by offering a comparison of the two conferences. The system also recognizes incomplete queries, such as ≪Compare IBM≫, and asks the user to provide a second organization for the comparison. When the second element is provided (e.g., ≪Hewlett-Packard≫), the system returns the results.

We implemented the *compare* query after performing the user study reported in Section VII with the aim of i) producing a concrete example of how to extend the chatbot with additional queries and ii) addressing the feedback of one of the users who requested this functionality. Hence, it is not covered in the evaluation. Section VIII presents the procedure we adopted for extending AIDA-Bot with the *compare* query, as an example for developers who want to add new query types to a conversational agent based on this architecture.

V. INTERACTION SCENARIOS

Three main scenarios can occur during the system interaction with the user. The first one is depicted in Figure 7. The user question is sent by the User Entity to the Chatbot

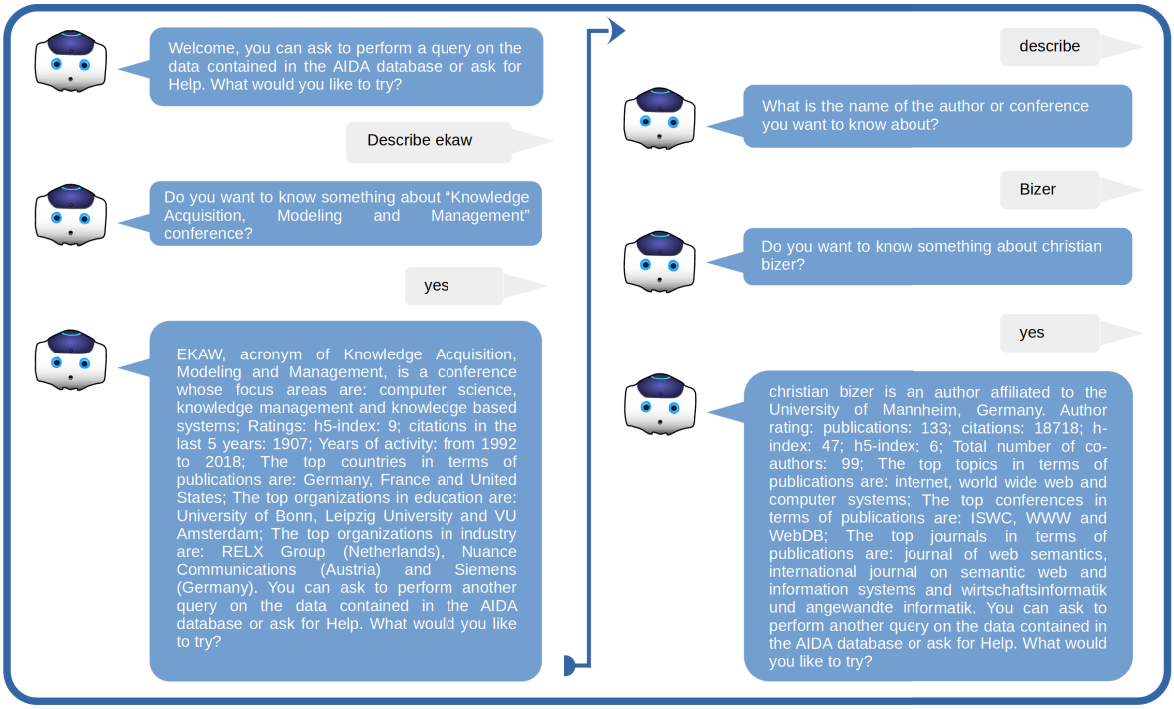


FIGURE 5. Example of interaction for the *describe* query.

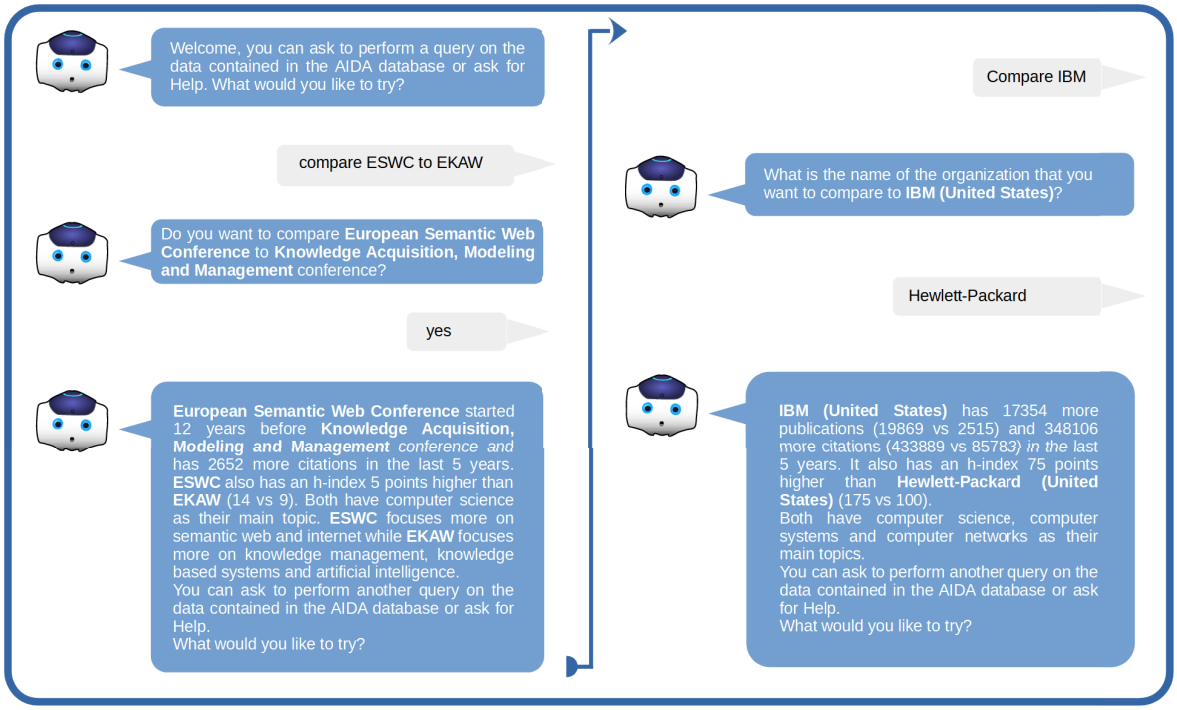


FIGURE 6. Example of interaction for the *compare* query.

Engine. For example, the user may ask: ≪Count the papers on machine learning≫. The Chatbot Engine sends it to the Parser, via the Web Server with a *Parser API request*. The Parser analyses it and extracts all the parameters necessary for formulating a query. In the case of the question ≪count the papers on machine learning≫, the instance value is '*machine*

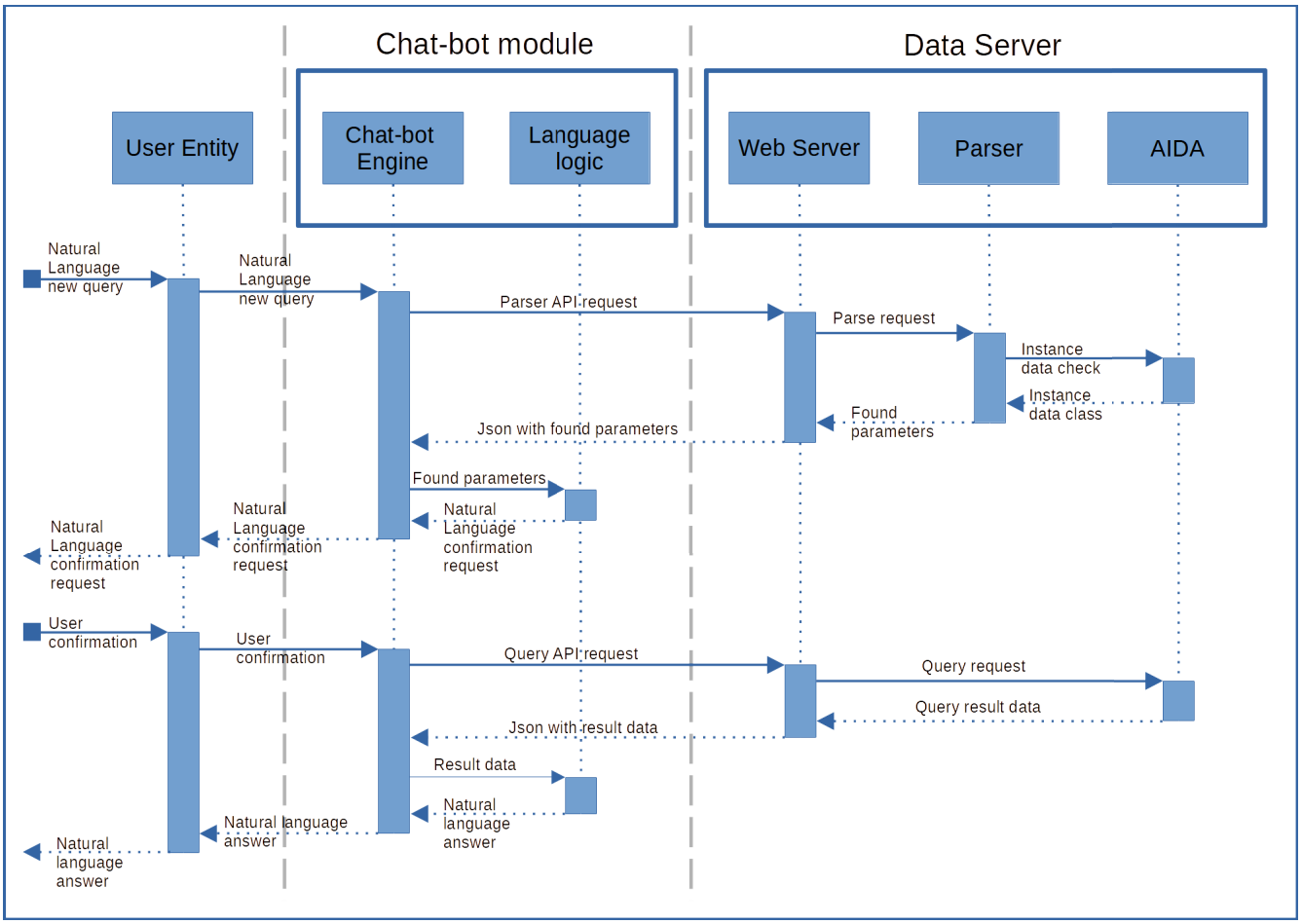


FIGURE 7. First scenario. The user question is complete and the system finds all the related information.

learning', the automatically recognised class is *'Topic'*, and the query-building parameters are the keyword *count* and the class *papers*. The Parser then returns to the Chatbot Engine a JSON containing all the parameters. Since in this case all the necessary parameters have been identified, the Chatbot Engine passes them to the Language logic module for generating a confirmation request to be sent to the user through the User Entity. For example, the chatbot may ask: ≪Do you want to know how many papers on machine learning topic are in AIDA?≫ When the user confirms that the interpretation is correct, the confirmation is transmitted by the User Entity to the Chatbot Engine, which sends to the Web Server a *query API request*. The latter then runs the query on the knowledge graphs (in this case AIDA KG) and sends the results to the Chatbot Engine as a JSON file. The results are then sent from the Chatbot Engine to the Language logic module to generate a natural language answer, which will be returned to the user through the User Entity.

In the second scenario, depicted in Figure 8, the flow is similar to the first one, except that the system fails to recognise the instance. This happens when the user question does not specify an instance value or when the relevant instance is not found in the knowledge graph. For instance, ≪count papers on quantum mechanics≫ would fall in this scenario, since AIDA, which mainly covers Computer Science, does not contain an entity named "quantum mechanics". In this case, the Chatbot Engine that receives the JSON information in which this parameter is missing, forwards the received parameters to the Language logic module so that it prepares a request, in natural language, which will be sent to the user through the User Entity. For instance, ≪I am sorry, I didn't understand. What is the name of the *topic*, *conference*, *organization*, or *author* whose papers I should count?≫ If a new and verifiable instance value is then entered by the user, it is transferred from the User Entity to the Chatbot Engine which will send to the Web Server a *find API request*. The Web Server will check the data with the AIDA module and will return the results to the Chatbot Engine. The latter will then perform a *query API request* to the Web Server and, as in the first scenario, will then return the results to the user in natural language.

In the third scenario (see Figure 9), the system fails to extract a query-building parameter from the user question. An example of a question missing query-building parameters is ≪list the top 5 papers on machine learning≫, where the user does not specify a query-building parameter of type

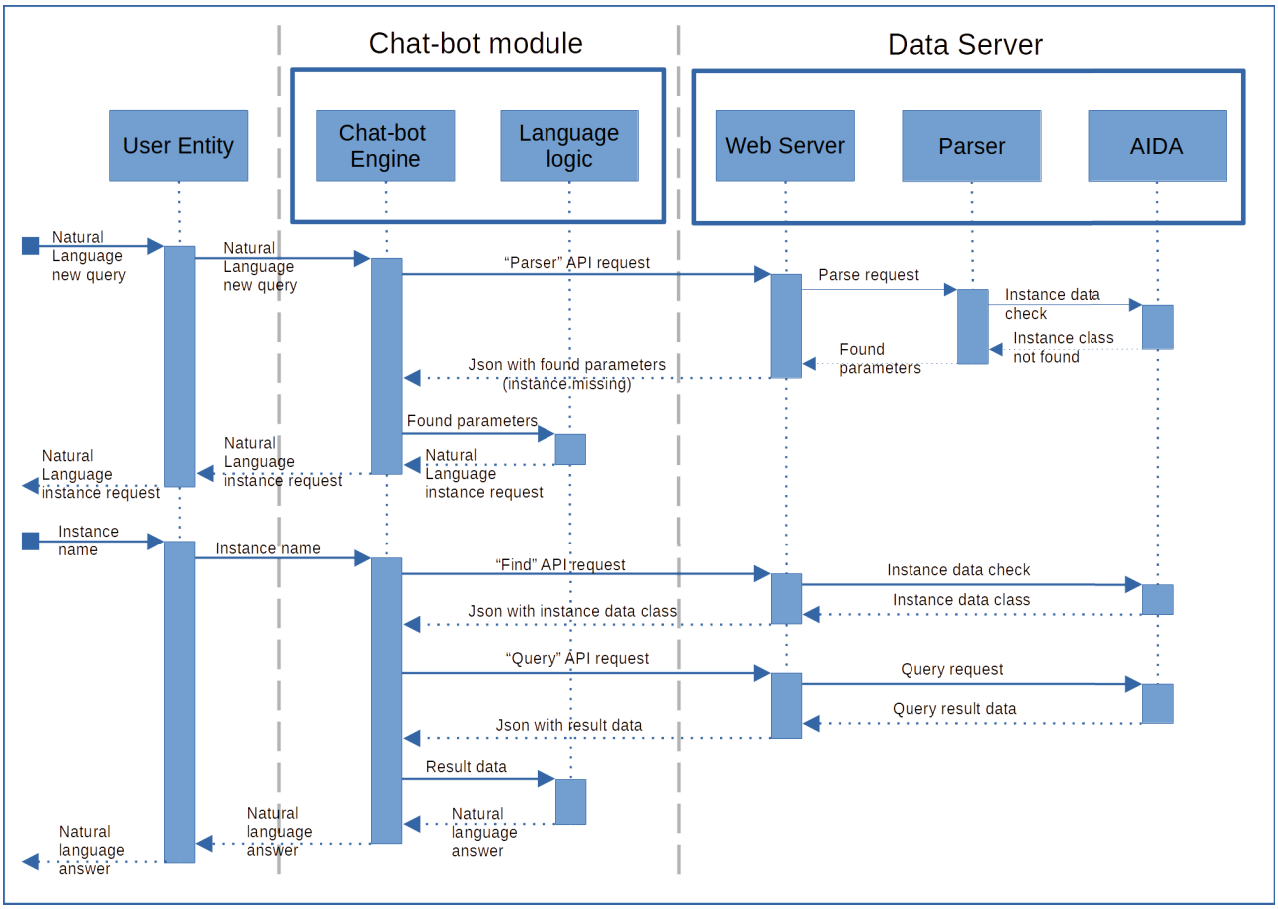


FIGURE 8. Second scenario. The chatbot cannot find an instance value in the user question.

"order", required by the *list* query. In this scenario, the Chatbot Engine will send the user, through the User Entity, a request in natural language, prepared by the Language logic module asking to specify the missing parameters. For instance, ≪Which sorting option do you prefer between: (1) citations and (2) citations in the last 5 years?≫ This time the user input is checked directly from the Chatbot Engine, which, as in the other scenarios, will perform a *query API request* to the Web Server, and will return the results to the user in natural language.

VI. THE USER ENTITIES

To demonstrate the versatility of this architecture, we implemented four different AIDA-Bot respectively using Alexa devices, web browsers, Telegram, and an NAO humanoid robot. In this section, we describe these prototypes in detail. We share the codebase of all our implementations to enable developers to easily re-implement and extend our architecture.

A. AMAZON ALEXA

We used the Alexa Skill Kit[34] to develop an Amazon Alexa skill. We chose the custom model for our skill and the Alexa-hosted option to host our skill backend resources (e.g., the AWS Lambda function we have employed). The Alexa-hosted choice uses Node.js as technology to develop the backend. Note that the use of the developed skill is limited to the user account owner until the application is approved by Amazon. This means that to test our application the reader needs to download the skill and upload it using a developer account. The skill code can be freely downloaded.[35]

Alexa provides all the necessary technology for speech recognition (speech-to-text) as well as speech synthesis (text-to-speech). Therefore, the sentence spoken by the user is first speech-to-texted and the resulting text is used as input for our application. Alexa's skill structure is divided into two different parts. The first one, called the *Voice Interaction Model*, includes utterances to define the different tasks (in Alexa known as *intents*). The parameters necessary for the execution of the intents are called *slots*. The second part, the *AWS Lambda*, contains all the necessary logic. The Voice Interaction Model and the AWS Lambda communicate through a JSON session containing the kind of intent (i.e. task) detected by the model, the values assigned to the slots, and the required results, once processed by the AWS Lambda. The AWS Lambda and the data server communicate through

[34]https://developer.amazon.com/en-US/alexa/alexa-skills-kit

[35]https://github.com/infovillasimius/aida.git

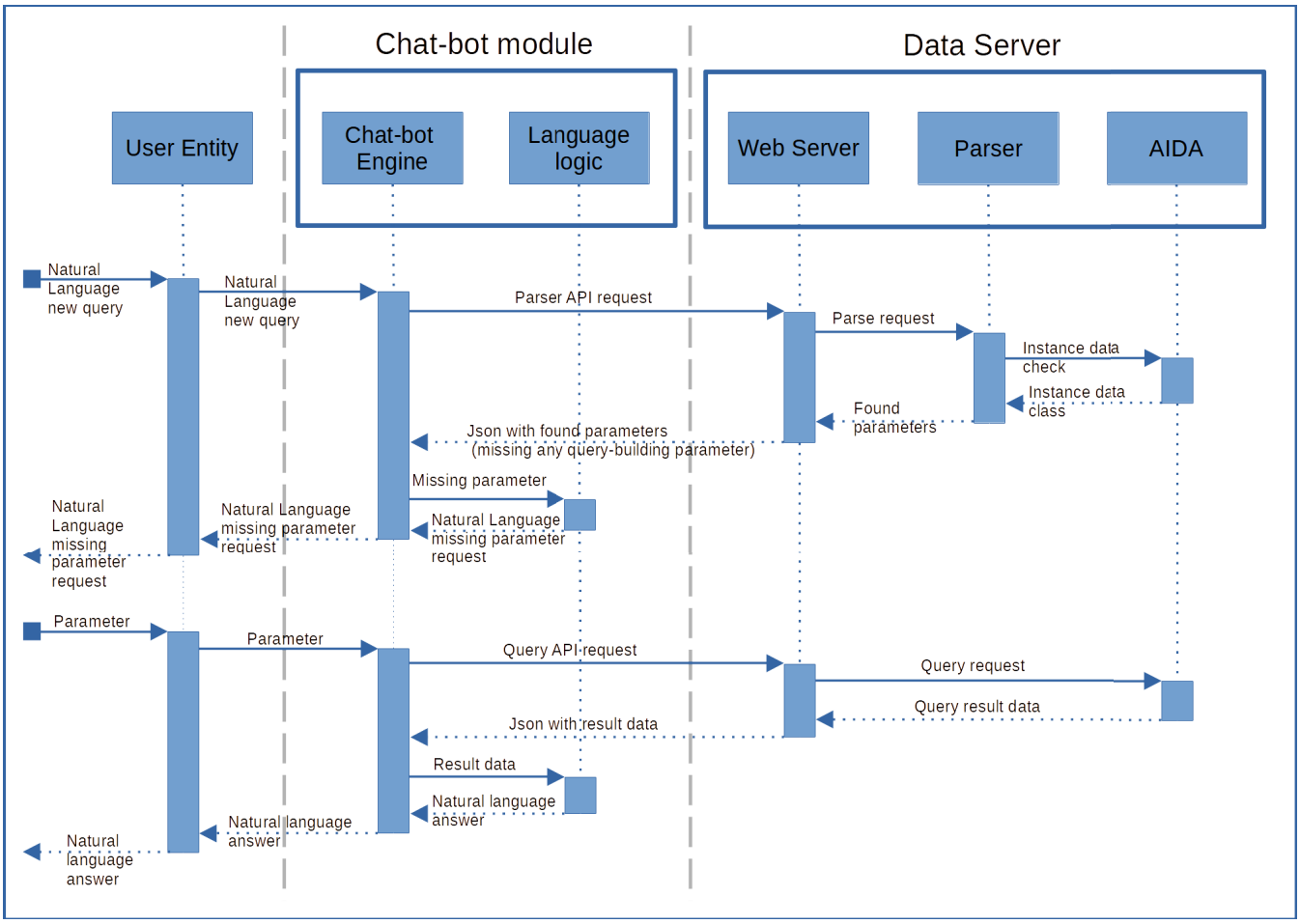


FIGURE 9. Third scenario. The chatbot cannot find a query-building parameter in the user question.

REST API calls. In the developed skill there are 9 intents (one for each developed query type and five Amazon default intents, like cancel, navigate home, and so on) and 7 slot types. Figure 10 shows the intent related to the **count query** previously mentioned. The sample utterances indicate three possible ways how this query can be enabled by natural language expressions spoken by the user. With the first option, *count*, the skill would enter in wizard mode and would ask single questions to the user to fill each slot of the intent. With the other two, *count {query}* and *how many {query}*, the user is allowed to formulate a complex question in natural language (starting with *count* or *how many*) to be parsed to identify the slots of the intent. If some of the slots cannot be filled (because not mentioned or not understood by the parser), the intent would enter in wizard mode and would ask the user ad-hoc queries for the slots that remain to be filled.

The developed skill can validate user inputs, limited to predetermined contents such as those with class *C*. We created custom slots, which do not allow dynamic control for content validation, and built-in slots (such as AMAZON.SearchQuery), which allow a user text to be validated using a more elaborated computation (e.g., the text is sent and checked against defined patterns, keywords, and values from the database). The Parser module, which resides on the data server, is responsible for this check and, as output, assigns the value of the slot (an instance in this case) to the correct class *I*. A video showing this user entity and how to install AIDA-Bot on Alexa is available here.[36]

B. THE WEB APPLICATION

The web application is developed in Javascript and JQuery, offering similar functionality as the Alexa skill mentioned above. The browser provides voice support as long as is compatible with the Web Speech API (e.g., Google Chrome). The session variables are kept in the browser memory and no data is permanently stored on the client's PC or smartphone. User requests are transmitted to the Data Server using REST API calls, as shown for the Alexa implementation. The difference with respect to the Alexa case is that the language logic and the Chatbot Engine (performed by the *voice interaction model* and the *AWS Lambda*) run through Javascript on the browser. Besides, the necessary technology, the *Asynchronous Speech Recognition and Text-to-Speech*

[36]How to import and test AidaBot Alexa Skill - https://youtu.be/7ANn_u-zX1Q

Intent

HowManyIntent

Sample Utterances

- count
- count {query}
- how many {query}

Intent Slots

Name	Slot Type	Type	Values	Use
subject	Aida_Count_Subject	Custom with values	authors, papers, conferences, organizations, citations	Used for the subject of the query (*TYPE-A* item)
object	Aida_Queryable_Object	Custom with values	topics, conferences, organizations, authors, papers	*Contains the class of the instance object (TYPE-B item)*
instance	Aida_Instance	Custom with values	no	Contains 'no' if no value is specified for the instance
item	Amazon.SearchQuery	Built-in	any	Used for input during wizard
query	Amazon.SearchQuery	Built-in	any	Sent to the parser module for extraction of values for the other slots

FIGURE 10. Alexa skill Intent for the count query.

components are provided by the Web Speech API.[37] The user is invited to play with the online demo which is available at https://w3id.org/aida/bot. The source code we developed can be freely downloaded.[38]

C. THE TELEGRAM APPLICATION

AIDA-Bot on Telegram is based on Python and its Chatbot Module runs on the Data server. The Chatbot Engine module through the Telegram API looks for incoming messages. When new messages arrive, the system processes them in order of arrival, and replies to them. Currently, the system can only process text messages. In future work, we plan to also include speech recognition modules. The Language logic module performs the same tasks as the implementations of the other user entities. The main difference with the implementation on the browser is that the session data are necessarily stored on the server side and are associated with the unique ID of the Telegram chat to which they refer. AIDA-Bot for Telegram is accessible from https://t.me/AidaChatBot. Figure 11 shows an example of an interaction between the Telegram bot and the user. In this example, the user is performing a count query asking how many papers there are with topic *machine learning*. The source code of the application can be freely downloaded from a public repository.[39]

D. THE HUMANOID ROBOT

In this section, we will first introduce the humanoid robot we have used as an additional user entity to interact with AIDA-Bot. Then, we will describe the human-robot interaction more in detail.

1) THE NAO ROBOT

The NAO robot[40] is controlled by the NAOqi OS, a specialized Linux-based operating system that has access to a wide set of tools and sensors to control the robot and interact with the users. NAO can be programmed and accessed by the Choregraphe suite.[41] Choregraphe is a platform for creating dialogues, behaviors, and animations, which can be tested directly on a real robot, or a simulated one. Within Choregraphe is also possible to easily create applications, that can include dialogs, services, and advanced behaviors using the graphical interface and without the need for coding. Moreover, advanced users can use Python and its libraries to extend the basic functionalities for more sophisticated human-robot interactions.

[37] https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
[38] https://github.com/infovillasimius/aidaBot
[39] https://github.com/infovillasimius/AidaTelegramBot
[40] https://www.softbankrobotics.com/emea/en/nao
[41] https://www.softbankrobotics.com/emea/en/support/nao-6/downloads-softwares

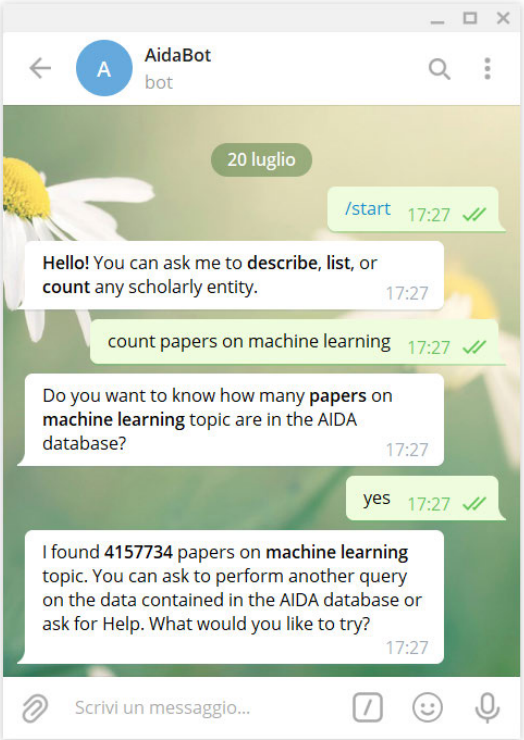


FIGURE 11. AIDA-Bot on telegram.

2) THE DEVELOPED HUMAN-ROBOT INTERACTION

The human-robot interaction has been developed using the Choregraphe suite, extending the basic functionalities in Python language. In particular, we have developed different behaviours of the robot when interacting with the user according to the defined system architecture shown in Figure 1. The human-robot interaction we developed is shown in the application logic diagram depicted within Figure 12. Its flow starts with the `Set Language` box on the top left corner used to select English as the robot's main language for speaking and for speech recognition. Right after, there is the `Init` box. It indicates that the robot moves to the init position, that is, it stands up from a rest position and welcomes the user. Under the hood, it sends an activation signal to the `AIDA-Bot` box, which enables the boxes responsible to make the robot to pronounce the welcome message and execute the related animation. At the end of the welcome animation, the robot is ready to process inputs thanks to the voice recognition module contained in the `Dialog` box. The task of this module is simply to forward the input text to the `AIDA-bot` box and send a stop signal to the `Nao bored` box. Each time the user says something, this box performs a speech-to-text conversion and sends the resulting text to the Chatbot module of Figure 1, contained in the `AIDA-Bot` box. As already shown in Section III, the Chatbot Engine and the Language Logic module will generate the appropriate answers to the user question by connecting, when needed, to the data server similarly as with the other user entities. Once the response is generated from the `AIDA-Bot` box, this is sent to `Say_Text` box which enables the robot to speak and say the response to the user. At the same time, a message is also sent to the `Motion` box, which chooses the correct animation for the robot to perform. There are eleven possible groups of animations which depend on the type of answer that the robot will have to give the user. For example, if the robot needs to ask a question to the user, it will gesticulate accordingly; if the robot says a statement it will perform a firm animation. If the robot's response includes a list of options from which the user must choose, it will use some sort of pointing animation. Each group contains a set of animations of the same kind. Once a group has been identified depending on the type of response to return to the user, the robot keeps performing random animations from the selected group until it ends to say the response to the user. To avoid overlaps between commands and movements, the `Dialog` box will be active only at the end of the animation cycle. When the robot finishes performing the animation associated with a specific answer, the flow goes to the `Stand` box and then to the `Dialog` box. Otherwise, if the user has not said anything else and the robot is thus ignored for a certain period, the `NAO bored` box takes control and performs one more animation of NAO representing a boring state. The goal is to randomly perform funny animations, showing that the robot gets bored if it is not interacting, with the aim of attracting the user's attention to hopefully resume the conversation. Two groups of animations are not contained in the `Motion` box and are run only once and at specific moments. The `Welcome` box animation is performed when the application starts whereas the `Goodbye` box animation is run when the user exits the application saying *quit* or *goodbye*. The `Goodbye` box animation is connected to the `Crouch` box, which puts NAO in a crouched position and stops the application.

A video showing the interaction between a user and a simulated NAO and how the flow of the described human-robot interaction is executed can be watched at https://youtu.be/ARS7RLggQwA. One more video showing the interaction between the user and the real robot can be watched at https://youtu.be/sZl1bw2-l2A. To make the system reproducible, the entire Choregraphe project and its related source code can be freely downloaded.[42]

VII. EVALUATION

In this section, we describe the user study we carried out for evaluating the usability of the AIDA-bot and its effectiveness in supporting users. Given that the four developed user entities (Amazon Alexa, Web Application, Telegram Application, and Humanoid Robot) share the same architecture, we decided to perform the user study on the Web Application, since it is the most accessible platform for all the users.

[42]https://github.com/infovillasimius/NaoAidaBot

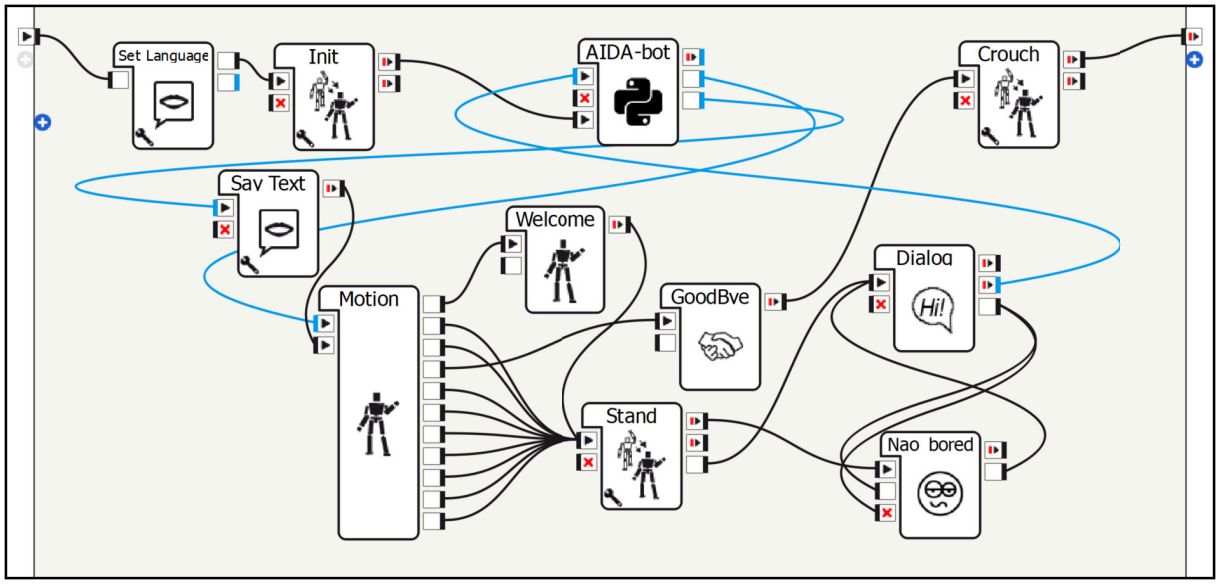


FIGURE 12. AIDA-Bot on NAO - logic scheme of the human-robot interaction.

A. USER STUDY

We conducted a user study on AIDA-Bot to assess the usability of the user interface as well as the quality and usefulness of the answers. In analogy with the purpose of the validation datasets of deep learning models, before showing the system to the annotators to collect their feedback, we showed the system to a single user. The idea was to fix simple errors/bugs in the system and to smooth the annotating process as much as we could using just one user as a validator. After that, we organized individual sessions with 15 researchers (different from the one chosen before) with expertise in Computer Science.

In each session, we first presented AIDA-Bot describing its functionalities for about 15 minutes. We then assigned the annotators the task of analyzing at least an author, an organization, and a conference within their research area to assess the quality of the resulting answers. After the hands-on session, the researchers filled out a four-section survey about the overall experience. The first section assessed the researchers' **user background** and expertise. The second section was a standard questionnaire on **System Usability Scale (SUS)** for assessing the usability of AIDA-Bot. The third section allowed the user to rate the **quality of the answers** for the chosen author, organization, and conference as well as for three types of queries (list, count, describe) on a Likert scale (from 1 to 5). As discussed in Section IV-D, the *compare* query was implemented only after this user study, hence it is not covered here.

The fourth and final section contained seven **open questions** about the strengths and the weaknesses of AIDA-Bot. The questions and the detailed results of the questionnaire are freely available at https://w3id.org/aida/downloads/AIDA-Bot-Chat-Evaluation.xlsx.

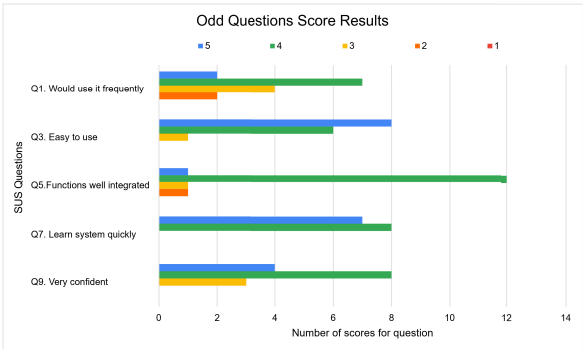


FIGURE 13. SUS questionnaire results for odd questions. The higher the score the better the system.

In the following sections, we will discuss the results more in detail.

1) USER BACKGROUND

The fifteen users were researchers in Computer Science at The Open University (UK), the University of Cagliari (IT), the National Council of Research (IT), the European Commission, Joint Research Centre (IT), the Sorbonne Paris North University (FR), and the Institute for Applied Informatics (DE). On average, they had 7.5 years of experience as a researcher. Four of them had at least 10 years of experience. Five of them had also experience in organising conferences, workshops, special issues, and similar events. Their areas of expertise included *Artificial Intelligence*, *Natural Language Processing*, *Digital Libraries*, *Semantic Web*, *Open Science*, *Digital Libraries*, *Blockchain*, *Robotics*, *Information Retrieval*, *Human-Computer Interaction*, *Computer Vision*, *Data Privacy*, *Sentiment Analysis*, *Knowledge Graphs*, *Data Engineering*, and *Symbolic and sub-symbolic AI*.

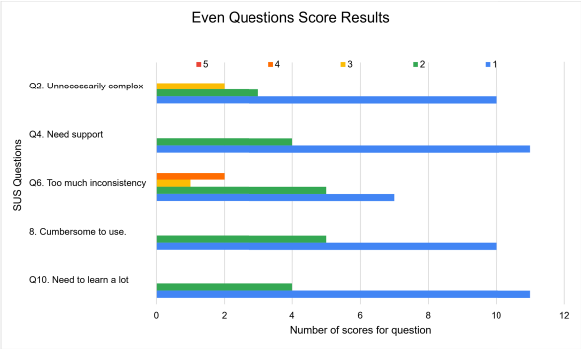


FIGURE 14. SUS questionnaire results for even questions. The lower the score the better the system.

TABLE 2. Quality assessment questions. The scores assess the quality of main queries (count, list, and describe) as well as information returned by the chatbot while investigating authors, conferences, and organizations.

	How effectively did AIDA-Bot answer the query?			*How effective was the interaction with AIDA-Bot when investigating authors, conferences, organizations?*		
User	Count	List	Describe	Authors	Confer.	Organiz.
1	4	4	4	4	4	4
2	4	5	5	3	5	5
3	5	5	4	4	5	5
4	4	4	4	4	5	4
5	5	5	5	5	5	5
6	4	5	4	3	5	4
7	5	5	5	5	5	5
8	4	4	5	5	4	5
9	5	5	5	5	4	4
10	4	5	5	4	5	5
11	4	4	4	4	4	4
12	4	4	4	4	4	4
13	5	4	5	4	5	5
14	5	5	5	5	5	4
15	3	3	3	2	3	3
AVG	4.3	4.5	4,5	4,1	4,5	4,4

2) SUS QUESTIONNAIRE

The SUS questionnaire[43] provided excellent results, scoring 83.2/100, which is equivalent to an A grade, placing the AIDA-Bot in the 95 percentile rank.[44]

Figures 13 and 14 show the score distribution of the users. Specifically, Figure 13 focuses on the odd questions (the positive ones, which should obtain a high score), while Figure 14 focuses on the even ones (the negative ones, which should obtain a low score). The users considered AIDA-Bot easy to use (with an average score of 4.47 ± 0.62[45]) and believed that its functions were well-integrated (3.87 ± 0.62). They claimed that it was not too complex to use (1.47 ± 0.72) and that they would not need any help to use it in the future (1.27 ± 0.44). In addition, the SUS results also reported that most of the users felt very confident when using AIDA-Bot (4.07 ± 0.68) and would be happy to use it frequently (3.60 ± 0.88).

[43]SUS Questionnaire Questions: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[44]Interpreting a SUS score - https://measuringu.com/interpret-sus-score/

[45]With the notation $X \pm Y$ we specify that X is the average score and Y the standard deviation.

3) QUALITY ASSESSMENT

The users were asked to rate several facets of AIDA-Bot on a Likert scale (1-5) and could also leave an additional comment.

They considered the quality of the interface to be particularly good (with an average score of 4.00 ± 0.73).

They also suggested some additional features, such as 1) supporting also visual analytics to better highlight the main elements of the responses, 2) clarifying better when the execution failed (e.g., any server error or the interaction failed due to problems with the structure of the question), 3) improving entity disambiguation, and 4) providing links to relevant web pages to convey additional information.

Table 2 reports the user ratings regarding the three main queries and the three main entities analyzed in the evaluation. The three queries, *count*, *list*, and *describe*, received an average score of, respectively, 4.30 ± 0.60, 4.57 ± 0.62, and 4.50 ± 0.62. The users suggested including additional details that are missing from the chatbot default answers, such as the year of publication when listing articles.

They also suggested referencing additional external sources (e.g., Google Scholar) when describing entities (e.g., authors).

The users considered the information regarding authors, conferences, and organizations as quite accurate, with an average score of respectively 4.10 ± 0.85, 4.50 ± 0.62, and 4.40 ± 0.61. The lower score regarding authors seems to be mainly due to the fact that AIDA does not always correctly disambiguate researchers, resulting in multiple versions of the same individual.

Therefore, analytics on authors are typically less accurate.

4) OPEN QUESTIONS

We devote this section to summarising the answers to the open questions of the questionnaire.

Q1. What are the main strengths of AIDA-Bot? Ten researchers pointed out that the main strengths of AIDA-Bot are its simplicity and usability. Two of them appreciated the capacity to easily summarise information about an entity, which typically can only be obtained by performing multiple queries on much more complex systems. Two other researchers mentioned that AIDA-Bot is very innovative, since, to the best of their knowledge, this is the first chatbot in this domain. Three of them appreciated its velocity and responsiveness.

Q2. What are the main weaknesses of AIDA-Bot? Six researchers suggested that they would want the ability to perform more complex queries, in particular by combining multiple filters. Four others would like to have more types of items to query to better investigate the domain. Other three researchers mentioned that the chatbot does not remember the context given by previous questions. They suggest that it would be useful to allow the users to further elaborate on previous questions, for example by adding filters or requesting a more comprehensive answer. Three of them

complained that the requests must necessarily have the required structure otherwise the system does not understand them. One researcher criticised the interaction with the chatbot as not very natural.

Q3. Can you think of any additional features to be included in AIDA-Bot? The researchers suggested: 1) adding more information about conferences (e.g., deadlines, venue for the year, rank), 2) synchronizing the results with personal devices (e.g., sending results via email or downloading results as CSV files), 3) linking the returned information to other databases or sources, 4) allowing to perform complex queries by combining the existing ones, and 5) adding a type of query to determine the dominant topic of a venue in a period (e.g., "What is the dominant topic of this year in Artificial Intelligence conferences?"), 6) the ability to combine multiple filters (e.g., listing all papers of a researcher in a given journal), 7) improving the natural language understanding, allowing users to ask questions in different ways, 8) adding more details and metrics about entities when answering to the describe query, and 9) automatically recommend new questions to keep the user engaged (e.g., after discussing the *AAAI* conference, asking if the user wants to know more about other conferences in *Artificial Intelligence*).

Q4. Can you think of any additional types of queries (other than count, list, and describe) for AIDA-Bot? A few users suggested the ability to filter each question by time period, e.g., asking for the top paper in Machine Learning between 2010 and 2020. One user suggested the ability to apply more filters in a single query, e.g., "List the top articles about Robotics at the International Semantic Web Conference". Another one asked to be able to compare venues and authors based on various ranks. One requested the ability to query also about academic *awards*, (e.g., best authors, best reviewer awards). Another user suggested supporting additional types of queries, such as ≪finding similar organisations≫, or ≪finding related papers≫. The same user suggested also producing a visual representation of the discussed metrics (e.g., charts of citations in time) whenever the version of the chatbot allows for it (e.g., Telegram).

Q5. What would you add to increase the accuracy/ comprehensiveness of the information returned by AIDA-Bot? Two researchers suggested improving the entity disambiguation systems, in particular for authors and organizations. Two researchers suggested developing a more complex GUI able to complement AIDA-Bot answers with additional visual aids (e.g., analytics, charts, scrollable lists). Four researchers suggested that the chatbot should be able to query multiple data sources and clarify the source of each piece of information. One researcher suggested leveraging state-of-the-art NLP techniques to support a larger number of user questions. Another one suggested including auto-completion when prompting organizations or topics, as this may improve the identification of these entities.

VIII. EXTENDING AIDA-BOT WITH A NEW QUERY TYPE

Thanks to the flexibility and modularity of the proposed architecture, adding new types of queries to the developed AIDA-Bot is straightforward. We must distinguish two scenarios for extending a conversational agent with a new query:

- If the system can reuse a previously defined REST API function for obtaining the required information from the knowledge graph, developers need only to update the communication and language management modules.
- Otherwise, developers need to also implement a new REST API function (and possibly update the knowledge graph with relevant new data).

In order to illustrate a concrete example, we discuss here the implementation of the query *compare*, which we introduced after it was suggested by a user in the evaluation (Section VII-A4). *Compare*, detailed in Sections IV-D, allows users to compare authors, conferences, and organizations. Since the relevant information can be easily obtained through the same REST API function used for the *describe* query, we are in the first case.

Therefore, the following items were added to the specified modules:

- *Chatbot Module - Chatbot Engine*. We added the keywords to identify the new query in the keyword control section. We also added a check for verifying the correctness of the user input and for the wizard to be activated in the event of errors or incomplete input from the user. For instance, if the two entities are not of the same type, the chatbot explains the issue to the user and asks them to correct the question.
- *Chatbot Module - Language logic*. We implemented 12 templates for handling questions and answers related to the new query (first entity request, second entity request, confirmation request, answers for too generic input for the first and second entities, answers for no results found for the first or the second entity, answer when the two entities are of different types, answer when the two entities correspond to the same instance, answer returning the comparison for entities of type author, conference, or organization). We also implemented a new function that compares the metrics of the two entities (e.g., number of papers, h-index) and provides relevant comments (e.g., "Yoshua Bengio has 340 more publications than Yann Lecun (608 vs 268)").

IX. CONCLUSION

In this work, we have introduced a new flexible approach for integrating conversational agents with knowledge graphs in the scholarly domain. On top of it, we developed AIDA-Bot, a prototype that leverages the Academia/Industry DynAmics (AIDA) Knowledge Graph for answering queries about research topics, research publications, authors, institutions, and publication venues. AIDA-Bot is to the best of our knowledge the first conversational agent of this kind.

This chatbot has been implemented in four different user entities: browsers, Alexa devices, Telegram clients, and a humanoid robot. Each of them allows the user to interact with the conversational agent which sends back to the user what has been asked. If needed, it asks the user to provide missing parameters according to a Natural Language Processing engine that translates the user questions in natural language into structured queries for the knowledge graph. Each of the prototypes is either publicly accessible or shown in a video. We also share the codebase of all the implementations to enable developers to reimplement and extend our architecture. Finally, a user study involving 15 computer scientists showed that the proposed solution is very usable and produces high-quality information.

We intend to sustain the project and extend its capabilities so that it may become a complete virtual assistant within the scholarly domain. More in detail, we would like to keep improving each of the four user entities. The three virtual implementations (i.e., browser, Telegram, Alexa) can be used by any researcher with Internet access. The user entity developed on top of a humanoid robot will be improved and brought to exhibitions and scientific conferences to let everyone interact with the robot and provide further feedback for our system. The reader notices that our architecture is general and can be applied in different other domains. As depicted in the schema of Figure 1, we would just need to load in the Data server a knowledge base from any domain and define the query in the Chatbot Module compliant to the data (as discussed in Section VIII). Last but not least, with the recent widespread of transformers, we obtained a very powerful technology to solve different tasks revolving within the domain of natural language processing, including question answering [75], [76]. However, they cannot scale to non-trivial databases nor answer set-based and aggregation queries. Based on these insights, our future direction will be to investigate how to best incorporate transformers in the proposed architecture and use them to answer complex queries on large-scale data.

REFERENCES

[1] A. Xu, Z. Liu, Y. Guo, V. Sinha, and R. Akkiraju, "A new chatbot for customer service on social media," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Denver, Colorado, May 2017, pp. 3506–3510.

[2] I. Chaoua, D. Recupero, S. Consoli, A. Harma, and R. Helaoui, "Detecting and tracking ongoing topics in psychotherapeutic conversations," in *Proc. 1st Joint Workshop AI Health*, vol. 2142, Stockholm, Sweden, 2018, pp. 97–108.

[3] L. Cui, S. Huang, F. Wei, C. Tan, C. Duan, and M. Zhou, "SuperAgent: A customer service chatbot for E-commerce websites," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics-Syst. Demonstrations*. Vancouver, BC, Canada: Association for Computational Linguistics, 2017, pp. 97–102.

[4] P. Shah, D. Hakkani-Tur, B. Liu, and G. Tur, "Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol., Volume*, 2018, pp. 41–51.

[5] N. Karatas, S. Tamura, M. Fushiki, and M. Okada, "Multi-party conversation of driving agents: The effects of overhearing information on lifelikeness and distraction," in *Proc. 6th Int. Conf. Human-Agent Interact.* New York, NY, USA: Association for Computing Machinery, Dec. 2018, pp. 84–91.

[6] A. Ait-Mlouk and L. Jiang, "KBot: A knowledge graph based ChatBot for natural language understanding over linked data," *IEEE Access*, vol. 8, pp. 149220–149230, 2020.

[7] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *ACM SIGKDD Explor. Newslett.*, vol. 19, no. 2, pp. 25–35, Dec. 2017, doi: 10.1145/3166054.3166058.

[8] H. Elbasri, A. Haddi, and H. Allali, "Improving E-learning by integrating a metacognitive agent," *Int. J. Elect. Comput. Eng.*, vol. 8, no. 5, p. 3359, Oct. 2018.

[9] V. Bellini, G. M. Biancofiore, T. Di Noia, E. D. Sciascio, F. Narducci, and C. Pomo, "GUapp: A conversational agent for job recommendation for the Italian public administration," in *Proc. IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*, May 2020, pp. 1–7. [Online]. Available: http://ceur-ws.org/Vol-2960/paper10.pdf

[10] A. Lommatzsch, "A next generation chatbot-framework for the public administration," in *Innovations for Community Services* (Communications in Computer and Information Science), M. Hodon, G. Eichler, C. Erfurth, and G. Fahrnberger, Eds. Zilina, Slovenia: Springer, 2018, pp. 127–141.

[11] C. van Noordt and G. Misuraca, "New wine in old bottles: Chatbots in government—Exploring the transformative impact of chatbots in public service delivery," in *Proc. 11th IFIP Int. Conf.*, 2019, pp. 49–59.

[12] Z. Callejas and D. Griol, "Conversational agents for mental health and wellbeing," in *Dialog Systems: A Perspective from Language, Logic and Computation*, T. Lopez-Soto, Ed. Cham, Switzerland: Springer, 2021, pp. 219–244.

[13] J. L. Z. Montenegro, C. A. Da Costa, and R. D. R. Righi, "Survey of conversational agents in health," *Exp. Syst. Appl.*, vol. 129, pp. 56–67, Sep. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419302283

[14] Y.-H. Liu, A. Arnold, G. Dupont, C. Kobus, and F. Lancelot, "Evaluation of conversational agents for aerospace domain," in *Proc. Joint Conf. Inf. Retr. Communities Eur.*, Jul. 2020, pp. 1–9. [Online]. Available: http://ceur-ws.org/Vol-2621/CIRCLE20_21.pdf

[15] A. Hogan, "Knowledge graphs," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–37, 2021.

[16] J. Bockhorst, D. Conathan, and G. M. Fung, "Knowledge graph-driven conversational agents," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2019, pp. 8–14.

[17] S. Peroni and D. Shotton, "OpenCitations, an infrastructure organization for open scholarship," *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 428–444, Feb. 2020.

[18] H. Wan, Y. Zhang, J. Zhang, and J. Tang, "AMiner: Search and mining of academic social networks," *Data Intelligence*, vol. 1, no. 1, pp. 58–76, Mar. 2019, doi: 10.1162/dint_a_00006.

[19] F. Weber, T. Wambsganss, D. Ruttimann, and M. Sollner, "Pedagogical agents for interactive learning: A taxonomy of conversational agents in education," in *Proc. ICIS*, 2021, pp. 1–17. [Online]. Available: https://aisel.aisnet.org/icis2021/diglearn_curricula/diglearn_curricula/13

[20] S. Angioni, A. Salatino, F. Osborne, D. R. Recupero, and E. Motta, "AIDA: A knowledge graph about research dynamics in academia and industry," *Quant. Sci. Stud.*, vol. 2, no. 4, pp. 1356–1398, 2021.

[21] A. A. Salatino, T. Thanapalasingam, A. Mannocci, A. Birukou, F. Osborne, and E. Motta, "The computer science ontology: A comprehensive automatically-generated taxonomy of research areas," *Data Intell.*, vol. 2, no. 3, pp. 379–416, Jul. 2020.

[22] A. Meloni, S. Angioni, A. Salatino, F. Osborne, and D. R. Recupero. (2021). *AIDA-Bot: A Conversational Agent to Explore Scholarly Knowledge Graphs*. [Online]. Available: http://ceur-ws.org/Vol-2980/paper310.pdf

[23] E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Mach. Learn. Appl.*, vol. 2, Dec. 2020, Art. no. 100006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827020300062

[24] K. Ramesh, S. Ravishankaran, A. Joshi, and K. Chandrasekaran, "A survey of design techniques for conversational agents," in *Proc. Int. Conf. Inf., Commun. Comput. Technol.* New Delhi, India: Springer, 2017, pp. 336–350.

[25] S. Hussain, O. A. Sianaki, and N. Ababneh, "A survey on conversational agents/chatbots classification and design techniques," in *Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl.* Matsue, Japan: Springer, 2019, pp. 946–956.

[26] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8689–8696.

[27] L. Li, K. Y. Lee, E. Emokpae, and S.-B. Yang, "What makes you continuously use chatbot services? Evidence from Chinese online travel agencies," *Electron. Markets*, vol. 31, no. 3, pp. 575–599, Sep. 2021, doi: 10.1007/s12525-020-00454-z.

[28] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y.-L. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics, Main Volume*, 2021, pp. 1–25.

[29] B. Borah, D. Pathak, P. Sarmah, B. Som, and S. Nandi, "Survey of textbased chatbot in perspective of recent technologies," in *Computational Intelligence, Communications, and Business Analytics*, J. K. Mandal, S. Mukhopadhyay, P. Dutta, and K. Dasgupta, Eds. Singapore: Springer, 2019, pp. 84–96.

[30] N. V. Karthik, "Comparative study on voice based chat bots," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 12, pp. 172–175, Dec. 2018.

[31] R. Alonso, E. Concas, and D. R. Recupero, "An abstraction layer exploiting voice assistant technologies for effective human—Robot interaction," *Appl. Sci.*, vol. 11, no. 19, p. 9165, Oct. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/19/9165

[32] T. Ammari, J. Kaye, J. Y. Tsai, and F. Bentley, "Music, search, and IoT: How people (really) use voice assistants," *ACM Trans. Comput. Hum. Interact.*, vol. 26, no. 3, pp. 1–17, 2019.

[33] J. Singh, M. H. Joesph, and K. B. A. Jabbar, "Rule-based chabot for student enquiries," *J. Phys., Conf.*, vol. 1228, no. 1, May 2019, Art. no. 012060. [Online]. Available: https://doi.org/10.1088/1742-6596/1228/1/012060

[34] S. Mohan and C. Chowdhary, "An Ai-based chatbot using deep learning," in *Intelligent Systems: Advances in Biometric Systems, Soft Computing, Image Processing, and Data Analytics*. London, U.K.: Apple Academic Press, 2019, ch. 12, pp. 231–242.

[35] L. Laranjo, A. G. Dunn, H. L. Tong, A. B. Kocaballi, J. Chen, R. Bashir, D. Surian, B. Gallego, F. Magrabi, A. Y. S. Lau, and E. Coiera, "Conversational agents in healthcare: A systematic review," *J. Amer. Med. Inform. Assoc.*, vol. 25, no. 9, pp. 1248–1258, Sep. 2018.

[36] C. W. Okonkwo and A. Ade-Ibijola, "Chatbots applications in education: A systematic review," *Comput. Educ., Artif. Intell.*, vol. 2, Mar. 2021, Art. no. 100033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666920X21000278

[37] R. Bavaresco, D. Silveira, E. Reis, J. Barbosa, R. Righi, C. Costa, R. Antunes, M. Gomes, C. Gatti, M. Vanzin, S. C. Junior, E. Silva, and C. Moreira, "Conversational agents in business: A systematic literature review and future research directions," *Comput. Sci. Rev.*, vol. 36, May 2020, Art. no. 100239.

[38] J. Brixey, R. Hoegen, W. Lan, J. Rusow, K. Singla, X. Yin, R. Artstein, and A. Leuski, "SHIHbot: A Facebook chatbot for sexual health information on HIV/AIDS," in *Proc. 18th Annu. SIGdial Meeting Discourse Dialogue*, 2017, pp. 370–373.

[39] L. Vaira, M. A. Bochicchio, M. Conte, F. M. Casaluci, and A. Melpignano, "MamaBot: A system based on ML and NLP for supporting women and families during pregnancy," in *Proc. 22nd Int. Database Eng. Appl. Symp.*, Villa San Giovanni, Italy, 2018, pp. 273–277.

[40] K.-J. Oh, D. Lee, B. Ko, and H.-J. Choi, "A chatbot for psychiatric counseling in mental healthcare service based on emotional dialogue analysis and sentence generation," in *Proc. 18th IEEE Int. Conf. Mobile Data Manag. (MDM)*, KAIST, Daejeon, May 2017, pp. 371–375.

[41] S. Divya, V. Indumathi, S. Ishwarya, M. Priyasankari, and S. K. Devi, "A self-diagnosis medical chatbot using artificial intelligence," *J. Web Develop. Web Designing*, vol. 3, no. 1, pp. 1–7, 2018.

[42] U. Bharti, D. Bajaj, H. Batra, S. Lalit, S. Lalit, and A. Gangwani, "MedBot: Conversational artificial intelligence powered chatbot for delivering tele-health after COVID-19," in *Proc. 5th Int. Conf. Commun. Electron. Syst. (ICCES)*, Budva, Montenegro, Jun. 2020, pp. 870–875.

[43] L. Ni, C. Lu, N. Liu, and J. Liu, "MANDY: Towards a smart primary care chatbot application," in *Proc. Int. Symp. Knowl. Syst. Sci.* Bangkok, Thailand: Springer, 2017, pp. 38–52.

[44] P. Thongyoo, P. Anantapanya, P. Jamsri, and S. Chotipant, "A personalized food recommendation chatbot system for diabetes patients," in *Cooperative Design, Visualization, and Engineering*, Y. Luo, Ed. Bangkok, Thailand: Springer, 2020, pp. 19–28.

[45] P. -T. Hsu, J. Zhao, K. Liao, T. Liu, and C. Wang, "AllergyBot: A chatbot technology intervention for young adults with food allergies dining out," in *Proc. CHI Conf. Extended Abstr. Human Factors Comput. Syst.*, Denver, Colorado, May 2017, pp. 74–79.

[46] M. Sarosa, M. Kusumawardani, A. Suyono, and M. H. Wijaya, "Developing a social media-based chatbot for English learning," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 732, no. 1, Jan. 2020, Art. no. 012074, doi: 10.1088/1757-899x/732/1/012074.

[47] D. Rooein, D. Bianchini, F. Leotta, M. Mecella, P. Paolini, and B. Pernici, "ACHAT-WF: Generating conversational agents for teaching business process models," *Softw. Syst. Model.*, vol. 21, no. 3, pp. 891–914, Jun. 2022, doi: 10.1007/s10270-021-00925-7.

[48] B. R. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related FAQs," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1525–1530.

[49] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "End-to-end task-completion neural dialogue systems," in *Proc. IJCNLP*, 2017, pp. 1–11.

[50] A. Følstad, T. Araujo, S. Papadopoulos, E. L.-C. Law, E. Luger, M. Goodwin, and P. B. Brandtzaeg, in *Proc. 5th Int. Workshop Chatbot Res. Design, CONVERSATIONS*, in Lecture Notes in Computer Science, vol. 13171, 2022, pp. 3–69. [Online]. Available: www.scopus.com

[51] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. N. Ngomo, "Survey on challenges of question answering in the semantic web," *Semantic Web*, vol. 8, no. 6, pp. 895–920, Aug. 2017.

[52] R. G. Athreya, A.-C. N. Ngomo, and R. Usbeck, "Enhancing community interactions with data-driven chatbots—The DBpedia chatbot," in *Proc. Companion Web Conf. Web Conf.*, Lyon, France, 2018, pp. 143–146.

[53] K. Stasaski and M. Hearst, "Semantic diversity in dialogue with natural language inference," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2022, pp. 85–98. [Online]. Available: https://aclanthology.org/2022.naacl-main.6

[54] J. Sreekantan, C. Hutchison, and P. Amatya, "Expert system for question answering on anomalous events and mitigation strategies using bidirectional transformers and knowledge graphs," in *Proc. ADIPEC*, Abu Dhabi, United Arab Emirates, 2022, pp. 881–891, doi: 10.2118/211855-MS.

[55] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, and S. Bowman, "QuALITY: Question answering with long input texts, yes!" in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2022, pp. 5336–5358. [Online]. Available: https://aclanthology.org/2022.naacl-main.391

[56] D. Fensel, U. Simsek, K. Angele, E. Huaman, E. Karle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler. (2020). *Knowledge Graphs Methodology, Tools and Selected Use Cases*. [Online]. Available: http://lib.ugent.be/catalog/ebk01:4100000010122122

[57] M. Mora-Cantallops, S. Sánchez-Alonso, and E. García-Barriocanal, "A systematic literature review on Wikidata," *Data Technol. Appl.*, vol. 53, no. 3, pp. 250–268, Sep. 2019.

[58] M. Nayyeri, G. M. Cil, S. Vahdati, F. Osborne, M. Rahman, S. Angioni, A. Salatino, D. R. Recupero, N. Vassilyeva, E. Motta, and J. Lehmann, "Trans4E: Link prediction on scholarly knowledge graphs," *Neurocomputing*, vol. 461, pp. 530–542, Oct. 2021.

[59] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, no. 3, pp. 489–508, 2017.

[60] F. Osborne and E. Motta, "Pragmatic ontology evolution: Reconciling user requirements and application performance," in *Proc. Int. Semantic Web Conf.* Monterey, CA, USA: Springer, 2018, pp. 495–512.

[61] S. R. Bader, I. Grangel-Gonzalez, P. Nanjappa, M.-E. Vidal, and M. Maleshkova, "A knowledge graph for Industry 4.0," in *Proc. Eur. Semantic Web Conf.* Herakleio, Greece: Springer, 2020, pp. 465–480.

[62] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 396–413, Feb. 2020.

[63] P. Manghi, C. Atzori, A. Bardi, M. Baglioni, J. Schirrwagen, H. Dimitropoulos, S. La Bruzzo, I. Foufoulas, A. Mannocci, M. Horst, and A. Czerniak, "OpenAIRE research graph dump," Tech. Rep., Nov. 2020. [Online]. Available: https://zenodo.org/record/7488618/export/hx#.ZAXTZZPP1uU, doi: 10.5281/zenodo.7488618.

[64] Y. Zhang, F. Zhang, P. Yao, and J. Tang, "Name disambiguation in AMiner: Clustering, maintenance, and human in the loop," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Jul. 2018, pp. 1002–1011.

[65] M. Y. Jaradeh, A. Oelen, K. E. Farfar, M. Prinz, J. D'Souza, G. Kismihók, M. Stocker, and S. Auer, "Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge," in *Proc. 10th Int. Conf. Knowl. Capture*. New York, NY, USA: Association for Computing Machinery, Sep. 2019, pp. 243–246.

[66] D. Dessi, F. Osborne, D. R. Recupero, D. Buscaldi, E. Motta, and H. Sack, "AI-KG: An automatically generated knowledge graph of artificial intelligence," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2020, pp. 127–143.

[67] M. Fenner and A. Aryani, "Introducing the PID graph," *Front Matter.*, Mar. 2019, doi: 10.5438/jwvf-8a66.

[68] M. Visser, N. J. van Eck, and L. Waltman, "Large-scale comparison of bibliographic data sources: Scopus, web of science, Dimensions, Crossref, and Microsoft Academic," *Quant. Sci. Stud.*, vol. 2, no. 1, pp. 20–41, 2021, doi: 10.1162/qss_a_00112.

[69] D. W. Hook, S. J. Porter, and C. Herzog, "Dimensions: Building context for search and evaluation," *Frontiers Res. Metrics Anal.*, vol. 3, p. 23, Aug. 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/frma.2018.00023

[70] S. Angioni, A. Salatino, F. Osborne, D. R. Recupero, and E. Motta, "Integrating knowledge graphs for analysing academia and industry dynamics," in *Proc. ADBIS, TPDL EDA Common Workshops Doctoral Consortium*. Cham, Switzerland: Springer, 2020, pp. 219–225.

[71] A. A. Salatino, T. Thanapalasingam, A. Mannocci, F. Osborne, and E. Motta, "The computer science ontology: A large-scale taxonomy of research areas," in *The Semantic Web—ISWC*, D. Vrandecic, K. Bontcheva, M. C. Suarez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, and E. Simperl, Eds. Monterey, CA, USA: Springer, 2018, pp. 187–205.

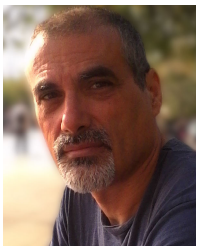
[72] W. Ammar, "Construction of the literature graph in semantic scholar," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol., Volume*, New Orleans, Louisiana, 2018, pp. 1–8.

[73] S. La Bruzzo, P. Manghi, and A. Mannocci, "OpenAIRE's DOIBoost—Boosting crossref for research," in *Digital Libraries: Supporting Open Science*, P. Manghi, L. Candela, and G. Silvello, Eds. Pisa, Italy: Springer, 2019, pp. 133–143.

[74] A. A. Salatino, F. Osborne, T. Thanapalasingam, and E. Motta, "The CSO classifier: Ontology-driven detection of research topics in scholarly articles," in *Digital Libraries for Open Knowledge*. Oslo, Norway: Springer, 2019, pp. 296–311.

[75] J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, and A. Halevy, "From natural language processing to neural databases," *Proc. VLDB Endowment*, vol. 14, no. 6, pp. 1033–1039, Feb. 2021, doi: 10.14778/3447689.3447706.

[76] J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, and A. Halevy, "Database reasoning over text," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 3091–3104.



ANTONELLO MELONI received the B.S. and M.S. degrees in computer science from the University of Cagliari, Italy, where he is currently pursuing the Ph.D. degree. He is the key developer of amr2fred, a tool for translating abstract meaning representation to motif-based linguistic knowledge graphs, and AIDA-bot. His research interests include natural language processing, human–robot interaction, conversational agents, and semantic web.

SIMONE ANGIONI received the B.S. and M.S. degrees in computer science from the University of Cagliari, Italy, where he is currently pursuing the Ph.D. degree. He is the Main Developer of the Academia/Industry DynAmics (AIDA) Knowledge Graph, an innovative resource for studying the relationship between academia and industry. His research interests include science of science, scientometrics, information extraction, semantic web, and robotics.

ANGELO SALATINO received the Ph.D. degree in methods for the early detection of research trends. He is a Research Associate with the Intelligence Systems and Data Science (ISDS) Group, Knowledge Media Institute (KMi), The Open University. In particular, his project aimed at identifying the emergence of new research topics at their embryonic stage. His research interests include semantic web, network science and knowledge discovery technologies, with a focus on the structures and evolution of science.

FRANCESCO OSBORNE is a Senior Research Fellow with the Knowledge Media Institute, The Open University, U.K., where he leads the Scholarly Data Mining Team. He is also an Assistant Professor with the University of Milano-Bicocca. His research interests include artificial intelligence, information extraction, knowledge graphs, science of science, and semantic web. He has authored more than a hundred peer-reviewed publications in top journals and conferences of these fields. He collaborates with major publishers, universities, and companies in the space of innovation for producing a variety of innovative services for supporting researchers, editors, and research polities makers. He has released many well-adopted resources, such as the computer science ontology and the computer science knowledge graph.

DIEGO REFORGIATO RECUPERO received the Ph.D. degree in computer science from the University of Naples Federico II, Italy, in 2004. From 2005 to 2008, he was a Postdoctoral Researcher with the University of Maryland, College Park, USA. He has been a Full Professor with the Department of Mathematics and Computer Science, University of Cagliari, Italy, since February 2022. He co-founded six companies within the ICT sector and is actively involved in European projects and research (with one of his companies, he won more than 40 FP7 and H2020 projects). His current research interests include sentiment analysis, semantic web, natural language processing, human–robot interaction, financial technology, and smart grid. He is the author of more than 190 conference papers and journal articles in these research fields, with more than 2400 citations. He received different awards in his career, such as the Marie Curie International Reintegration Grant, the Marie Curie Innovative Training Network, the Best Researcher Award from the University of Catania, the Computer World Horizon Award, the Telecom Working Capital, the Startup Weekend, and the Best Paper Award.

ENRICO MOTTA received the Laurea degree in computer science from the University of Pisa, Italy, and the Ph.D. degree in artificial intelligence from The Open University. He is a Professor with Knowledge Technologies and the former Director of the Knowledge Media Institute (KMi), The Open University, U.K., from 2000 to 2007. His research interests include the intersection of large-scale data integration and modeling, semantic and language technologies, intelligent systems, and human–computer interaction. He has led KMi's contribution to numerous high-profile projects, receiving over B#10.4M in external funding from a variety of institutional funding bodies and commercial organizations, since 2000.

Open Access funding provided by 'Università degli Studi di Cagliari' within the CRUI CARE Agreement