IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 June 2024, accepted 2 July 2024, date of publication 4 July 2024, date of current version 12 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3423359

RESEARCH ARTICLE

A Novel Knowledge Plug-In for Incorporating Information About Employability From the O*NET Database Into Conversational Agents

RUBÉN ALONSO [1,2], DANILO DESSÍ [3], ANTONELLO MELONI [4], AND DIEGO REFORGIATO RECUPERO [4]

[1]R2M Solution s.r.l., 27100 Pavia, Italy
[2]Programa de Doctorado, Centro de Automática y Robótica, Universidad Politécnica de Madrid-CSIC, 28006 Madrid, Spain
[3]GESIS Leibniz Institute for the Social Sciences, 50667 Cologne, Germany
[4]Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy

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

This work was supported by the H2020 Project STAR—Novel AI Technology for Dynamic and Unpredictable Manufacturing Environments under Grant 956573.

ABSTRACT The labor market is a dynamic and rapidly evolving environment. Job positions that require expertise in various sectors often lead candidates to question their suitability. Therefore, it is crucial to furnish them with relevant, accurate, and timely information. In this article, we introduce a knowledge plug-in for existing conversational agents designed to address specific queries related to the job market domain. Additionally, we propose an innovative method for dynamically creating diverse grammars and semantically associating users' questions with a predefined list of domain questions. We present a novel scheme to effectively tackle question-answering tasks in settings with limited resources. Our architecture relies on question-understanding and response-generation modules, both powered by Transformers, the O*NET occupational database, and recommendation engines that suggest training materials. Furthermore, we conducted a user study based on the System Usability Scale (SUS) score, revealing that users highly appreciated the proposed tool. This sentiment was particularly evident when the tool was integrated with other artificial intelligence chatbots capable of handling general information. Simultaneously, our engine adeptly manages information within the investigated domain, providing precise responses and recommendations. This work addresses a critical gap in the delivery of employment information and paves the way for the development of diverse functionalities to assist both candidates and employers.

INDEX TERMS Labor market, information extraction, chatbots, virtual assistant, user experience, human–robot interaction, online enrolling process.

I. INTRODUCTION

The job market, also referred to as the labor market, serves as the arena where employers seek employees and vice versa. It is not a physical location but rather a conceptual space illustrating the competition and interaction among diverse labor forces. The dynamics of the job market hinge significantly on the demand for labor and the availability of workers within the broader economy [1].

A recent survey reveals a noteworthy increase in hires since 2021, projecting a growth of 8.3 million jobs from 2021 to 2031.[1] Furthermore, according to the Bureau of Labor Statistics, the job market is tightening, evidenced by a rising number of quits or voluntary separations

The associate editor coordinating the review of this manuscript and approving it for publication was Maria Chiara Caschera.

[1]Source: Bureau of Labor Statistics https://www.bls.gov/emp/.

year-over-year. This trend suggests that employees are leaving their current positions in pursuit of better opportunities, contributing to increased competition in the market.[2] However, this competition is different from that of recent years. Instead of numerous candidates vying for limited positions, there is now a surplus of job openings with insufficient candidates to fill them. This issue can be attributed to the rapid pace of technological advancements [2], [3].

This situation can be challenging for applicants who might miss out on excellent opportunities due to a lack of specific skills or abilities in their resumes, which would otherwise qualify them for these positions. Society's evolving needs lead to fluctuations in the demand for various professions. Job seekers should adapt their skills to align with emerging positions, ensuring they secure well-paying jobs in growing industries with advancement potential.

CareerBuilder,[3] an employment website established in 1995 and operating across 23 countries with a presence in over 60 markets, offers labor market information, talent management software, and various recruitment-related services. The platform has identified several industries and jobs on the rise since 2021, spanning domains such as recreation, arts, renewable energy, personal services, healthcare, and information technology. Examples of growing roles within these domains include motion picture projectionists, ushers, lobby attendants, restaurant cooks, costume attendants, wind turbine service technicians, solar photovoltaic installers, logisticians, fitness trainers, animal caretakers, manicurists, nurse practitioners, occupational therapy assistants, physical therapy assistants, information security analysts, and software developers.

While various courses, lectures, and degree programs impart skills at different levels, the rapidly changing job market poses a challenge. Job-seekers with earned degrees may not possess all the required skills for their targeted positions due to market evolution. Consequently, they must stay abreast of demanding requirements by pursuing additional degrees or acquiring new skills through study or practice. By gaining these competencies, job-seekers increase their chances of being selected for their desired positions.

There are numerous online systems available for uploading or creating resumes and automatically searching for the most suitable jobs. Examples of these systems include Monster.com,[4] OfferZen,[5] LinkedIn,[6] CareerBuilder, Glassdoor,[7] JobStreet,[8] and ZipRecruiter,[9] among others. These platforms house millions of job offers and resumes from potential candidates.

[2]https://www.brainzmagazine.com/post/the-great-resignation-phenomena-employees-need-more-from-companies-to-stay
[3]https://www.careerbuilder.com/
[4]https://www.monster.com/
[5]https://www.offerzen.com
[6]https://www.linkedin.com/
[7]https://www.glassdoor.com/
[8]https://www.jobstreet.com/
[9]https://www.ziprecruiter.co.uk

While it is convenient to search for jobs using these systems, securing the best opportunities is not straightforward. For instance, if a position requires a computer science expert with fundraising skills, a candidate possessing only computer science skills may not be selected due to the absence of crucial elements in their resume, such as proposal writing experience or startup development.

To address this challenge, candidates need to recognize the importance of specific skills, like proposal writing in the mentioned example, for fulfilling job requirements. Obtaining precise information about such skills and where to acquire them is not always readily available. Additionally, providing accurate responses to such questions relies on data from trustworthy organizations that consistently update their information. Furthermore, job-related information should be mapped to values and normalized for effective querying with a certain degree of granularity.

Frequently, studies conducted at a younger age may not be directly applicable due to the dynamic nature of the job market and the emergence of new abilities sought by employers. Sometimes, required skills are acquired through years of work experience, while in other cases, investing a few months in studying a particular subject from a book, a short online course, or a seminar may suffice.

The most significant challenge lies in locating information effectively and consistently. A recent solution to this challenge is represented by conversational agents, which have witnessed substantial technological advancements [4]. The recent release of ChatGPT[10] has demonstrated impressive results in conversations across different domains. Despite claims that ChatGPT can answer follow-up questions, admit mistakes, challenge incorrect premises, and reject inappropriate requests, it often generates responses that are inaccurate and may mislead the user.

OpenAI, the research and deployment company behind ChatGPT, acknowledges certain limitations: i) ChatGPT may produce plausible-sounding but nonsensical or incorrect answers; ii) it is sensitive to tweaks in input phrasing or repeating the same prompt multiple times; iii) it tends to be excessively verbose and overuses certain phrases; iv) it does not ask clarifying questions when faced with ambiguous user queries; v) it may exhibit biased behavior when responding to inappropriate requests. Additionally, it is important to note that ChatGPT's APIs are not freely accessible.

As an illustrative example, if one were to inquire multiple times about the top *n* important skills required for a specific job, ChatGPT would consistently provide dissimilar responses, with skills varying or presented in a different order.

This issue has been further substantiated by the authors in [5], where they highlighted similar concerns with ChatGPT within the scholarly domain. In such instances, ChatGPT generated responses containing partially invented elements, leaving users without clear indications of correctness or errors.

[10]https://openai.com/blog/chatgpt/

While ChatGPT may provide accurate responses at times, complete reliance on it is not advisable, and alternative mechanisms should be explored. Similar challenges are observed in other large language models (LLM), such as LLaMa,[11] Koala,[12] and Alpaca.[13] Unlike ChatGPT, these models are downloadable and installable but demand substantial hardware resources.

To address the aforementioned challenges, needs, and limitations of ChatGPT, aligning with the objectives of the H2020 project STAR,[14] where the authors are actively involved, and following the similar goals outlined in [5], we propose a knowledge plug-in for existing conversational agents. This plug-in is designed to respond deterministically to specific questions within the job market domain, aiming to offer a reliable and trustworthy solution.

The knowledge base we have utilized is O*NET,[15] developed by the U.S. Department of Labor, Employment and Training Administration. It constitutes a vast data repository containing an extensive array of variables describing work and worker characteristics, including skill requirements. With over 20 years since its initial version, O*NET undergoes periodic updates and follows a well-defined data collection program encompassing both workers and employers. This program serves as a crucial source for training programs and labor studies in numerous countries, featuring a comprehensive taxonomy that defines occupations worldwide.

Each occupation within O*NET is linked to entities such as abilities, tools, work activities, etc., necessary for that specific job. Entities include levels and scores, aiding in the identification of the most critical ones for a particular occupation. As the most relevant and primary source of occupational information in the United States, O*NET has been successfully employed by authors in [6], introducing a novel method to identify skills from candidates' resumes and match them with job descriptions. Additionally, it has been utilized in [7], where a general architecture based on NLP technology was employed to provide recommendations for online job searches and skills acquisition.

Our tool is designed to respond deterministically to actions/questions related to the labor domain, utilizing the knowledge provided by O*NET. Examples include:

- List some alternative names for the job of Computer Scientist;
- List the jobs related to Software Engineering;
- List the top five skills required by a Data Engineer;
- What is the description of a Mechanical Engineer?
- Find the similarities between a Mechanical Engineer and a Civil Engineer;
- List the jobs that require Microsoft Office as a technical skill;
- Recommend me some material about social perceptiveness.

While Retriever Augmented Generation (RAG) [8] represents a significant advancement in expanding LLMs to incorporate specific knowledge, it is not immune to the issue of hallucination. Conversely, the responses provided by our tool are deterministic and are aligned with the information contained in O*NET. In more detail, the contributions we present in this paper are as follows:

- We define an architecture for the proposed knowledge plug-in that can be embedded in existing conversational agents.
- We propose a lightweight, innovative way to semantically associate the user's question with the most similar among a list of predefined ones, creating dynamically different grammars and exploiting Deep Learning and Natural Language Processing techniques.
- We introduce a flexible and efficient scheme to successfully solve a question-answering task where the responses are matched against a taxonomy. This taxonomy contains several types of entities for a certain job, each with a different score for different jobs. This allows us to exploit the ordering of entities for certain types of queries (e.g., "list the top five abilities for a computer scientist").
- We leverage the O*NET taxonomy with its entities, scores, and knowledge, enabling us to obtain information from O*NET and develop counting capabilities for our system (e.g., a query such as "list the top 3 skills currently required by a Sales Manager" can be formulated and correctly answered).
- We release a proof of concept[16] featuring an online web chatbot that utilizes the knowledge plug-in and is capable of responding to various user questions within the labor market domain.
- We provide a user study of the knowledge plug-in with an implemented chatbot, along with a questionnaire assigned to 15 laborers working in different companies. One part has been used to calculate the SUS score, and the second part has been designed to let users suggest feedback and improvements to the overall interaction.
- We present a couple of scenarios of applications of the proposed knowledge plug-in.
- We publicly release the code and the results of the annotators in a GitHub repository.[17]

The remainder of this paper is organized as follows. Section II discusses related works on conversational agents within the domain of the labor market. Section III includes the description of the O*NET database that we have used in this paper. Section IV introduces the architecture we have defined and details each constituent module. The user study we have performed is contained in Section V, where

[11] https://ai.meta.com/llama/
[12] https://bair.berkeley.edu/blog/2023/04/03/koala/
[13] https://crfm.stanford.edu/2023/03/13/alpaca.html
[14] https://cordis.europa.eu/project/id/956573
[15] https://www.onetcenter.org/database.html individual-files
[16] Knowledge plug-in proof of concept: http://hrilabdemo.ddns.net:8000/bot/.
[17] https://gitlab.com/hri_lab1/agent

we also include information about the annotators and their feedback. Section VI contains all the improvements we have made to address the comments of the annotators during their evaluation. Section VII describes two scenarios where users interacted with our system to find important information. Finally, Section VIII concludes the paper with findings and future directions.

II. RELATED WORK

The work proposed in this paper focuses on integrating artificial intelligence tools into conversational agents for deployment in real-world scenarios related to the job market and incorporating domain-specific knowledge sources into such technologies. Initiatives supporting job seekers in finding employment or enhancing their qualifications, as well as assisting employers in filling open positions through innovative technologies, have been explored in recent scientific literature [9]. Collaborative-filtering [10], similarity-based [11], and knowledge-based [12] recommendation systems, along with their combinations, represent common approaches in this domain. For instance, in [13], a hybrid recommendation system framework was proposed for job seekers. This framework matches candidates' resume content using a content-based similarity approach and incorporates online user interactions among job seekers and job posts through a graph representation. In [14], the authors introduced a Deep Semantic Structure Modelling (DSSM) approach to represent resumes and job postings, constructing a semantic embedding space. They utilized cosine similarity to determine the relationship between skills and job titles in this embedding space for recommendations. However, a limitation of their work is the oversight of considering which skills are essential for specific jobs, information that is typically known a priori for well-established job positions and could enhance the embedding space representation. Similarly, in [15], the joint representation of resumes and jobs in an embedding space was explored using convolutional neural networks (CNNs). Meanwhile, in [16], a word-level joint semantic space was proposed using recurrent neural networks (RNNs) and hierarchical ability-aware attention mechanisms to learn latent features of job abilities and candidates' resumes. Elements within this joint semantic space were utilized to train a fully connected network for predicting matching labels via a logistic function.

Today, current state-of-the-art approaches to rank candidates based on their skills and present them to employers are utilized by marketplaces such as LinkedIn, Monster.com, Fiverr,[18] OfferZen, and others, marking a significant rise in online hiring over the past decade. For instance, LinkedIn researchers outlined their method in [17], using latent representations and deep learning models to represent job requirements and candidate skills in the recruitment process. These latent representations match a recruiter's query (e.g., a candidate profile describing a person qualified for a specific open position) with LinkedIn members, generating a ranking to identify suitable candidates for the open position. Despite advancements in online hiring platforms, a key limitation of existing solutions is their inability to provide users with easy access to complex taxonomies, preventing them from discovering which skills would qualify them for specific job positions. Users can only find jobs fitting their profiles, without the ability to work on self-improvement aligned with job market dynamics and acquire new abilities that may be required. Our proposed architecture addresses this gap, allowing access to structured data through a Natural Language Interface (NLI) and advanced NLIDB technologies. Specifically, the proposed knowledge plug-in facilitates user interaction with the content of the O*NET database using natural language [18]. In the literature, NLIDBs have been designed using various technologies, including rule-based matching [19], and more recently, deep learning technologies [20], [21], [22]. Our solution utilizes advanced models based on the transformer architecture to detect text spans representing valuable information, which is then matched against entities in O*NET. However, our approach differs from existing ones, as a classic NLIDB typically focuses on converting natural language into SQL queries, whereas our approach uses SQL solely to retrieve entities from the O*NET database. The selection and refinement of the returned result are performed by semantically matching the entities of the database with the concepts detected in the user input.

One more crucial aspect concerning the use and design of such architectures is how users interact with them and perceive the delivered results. In response to this need, various works have concentrated on presenting generated results to users using diverse formats, including text or images. They design efficient implementations to accelerate latency or employ advanced conversational agents capable of delivering results and providing insights on the reasoning behind those results [23]. Users can seek explanations from the conversational agent, fostering a better understanding of the outcomes. Recently, conversational agents have found applications in various domains such as lifestyle [24], mental health [25], and customer support [26]. Within the job market context, conversational agent applications cater to a multitude of activities. In [27], the authors introduce CASExplorer, a conversational agent aiding individuals seeking enrollment in a university to choose the right major. It supports them during the decision process and exploration of required skills and job career opportunities. Leveraging users' strengths and personality traits, the system is based on the IBM Watson[19] suite. In [28], a text-based conversational agent was studied in various scenarios, including job interviews for entry-level job applicants. The authors demonstrated that AI through the conversational agent can enhance the efficiency and inclusivity of the recruitment process, with users naturally interacting with it. Therefore, conversational

[18]https://www.fiverr.com/

[19]https://www.ibm.com/watson

agents enable a proper evaluation of candidates, especially in high-risk scenarios where trust in the agent is essential. In [29], the authors proposed GUApp, a conversational agent allowing users to search for a job in a specific geographical area using content from the *Gazzetta Ufficiale*, the official journal of record of the Italian public administration. GUApp recognizes real-world entities from user questions and exploits a knowledge graph built by querying DBpedia[20] and crawling information about jobs from the ISTAT website,[21] a provider of profession taxonomies organized by sectors. Recommendations are based on users' skills, wishes, and ambitions, which are matched against the information contained in the knowledge graph.

RAG stands as a groundbreaking AI framework designed to enhance LLMs by seamlessly integrating external knowledge bases [8]. This integration serves a dual purpose: to ground LLMs in the most precise and current information available, while also providing users with insights into the generative process of these models. By implementing RAG within an LLM-based question-answering system, the model can access the latest, trustworthy facts and the users are granted visibility into the model's sources, thereby allowing them to verify the accuracy of its claims and build trust. Moreover, by anchoring an LLM with a pool of external, verifiable facts, the model's reliance on pre-embedded information within its parameters is notably reduced. Although this diminishes the generation of erroneous or misleading information, it still may generate hallucinated text.

In line with recent advancements, we propose a knowledge plug-in that can seamlessly integrate into conversational agents, associating user questions with a set of prepared questions through an innovative NLP schema. Subsequently, these associations are utilized via advanced latent representations to match skills and qualifications necessary for a searched job. This facilitates easy access to the knowledge contained in the O*NET database, a cornerstone for the US job market, supporting users in exploring skills and requirements for desired job positions. The reader notices that the responses provided by the proposed plug-in do not suffer from hallucinations. These responses are derived through deterministic computation, achieved by querying the O*NET database. Hence, a comparison between the two strategies would not be fair. Alternatively, our proposed system can be seen as a lightweight interface for browsing O*NET data.

Finally, the innovations introduced in this paper strategically align with EU projects related to the manufacturing sector (KITT4SME,[22] DENiM,[23]) focusing on the application of artificial intelligence to enhance activities related to the upskilling of workers and the provision of training recommendations. For instance, in the STAR[24] project, there is a specific task for the study of workers' training and continuous learning. This task involves analyzing the training needs of workers, assisting them in finding information related to their jobs and potential career paths, and recommending both project-generated and publicly available training materials. Chatbot extensions powered by well-known occupational databases seamlessly fit into the context of these projects, allowing upskilling to be addressed in more innovative ways.

III. THE O NET DATABASE

The primary source of information about employment in the United States is the O*NET Program.[25] It is designed to identify how rapidly jobs change and their impact on the economy and the workforce. One of its main outcomes is the O*NET database, developed under a Creative Commons license and freely available.[26] It consists of hundreds of standardized descriptions related to approximately 1000 US occupations. The project is active, and the database is continuously updated by various organizations. Millions of individuals use it to seek jobs and identify the required training for eligibility for certain jobs, and employers use it to pinpoint skilled professionals.

Each job in the O*NET database is associated with a set of skills, abilities, and knowledge and is characterized by various tasks and activities. The version of the database we utilized is 26.2, including 35 *Skills*, 33 *Knowledge*, 52 *Abilities*, 41 *Work Activities*, 4,127 *Tools Used*, 17,975 *Task Ratings*, and 8761 *Technology Skills*. We will refer to all of these as O*NET entities in the remainder of this paper. They are related to 1,016 occupations included in the O*NET database. Each of these occupations has a title and a related description. For instance, the job *Computer Systems Engineers/Architects* has the description *Design and develop solutions to complex applications problems, system administration issues, or network concerns. Perform systems management and integration functions*.

Skills represent the required expertise and are scored on a continuous range of [1-6]. A higher score indicates greater importance for the underlying job. For instance, *Time Management* is associated with the job *Information Technology Project Managers* with a score value of 4. The examples provided in this section refer to this occupation.

Knowledge encompasses the area of expertise for a specific occupation and is scored on a continuous range of [1-7]. For example, *Customer and Personal Service* has a score of 4.24.

Abilities denote a set of capacities for each occupation. Similar to *Knowledge*, *Skills*, and *Work Activities*, the same ability can be present in multiple jobs, with each occurrence associated with a different score. *Deductive Reasoning*, for instance, is an ability for the occupations *Information Technology Project Managers* and *Mathematicians*, with score values of 3.88 and 4.12, respectively.

Coordinating the Work and Activities of Others is an example of *Work Activities*, with scores ranging in the

[20]https://www.dbpedia.org/
[21]https://www.istat.it/en/
[22]https://kitt4sme.eu/
[23]https://denim-fof.eu/
[24]https://star-ai.eu/
[25]https://www.onetcenter.org/
[26]https://www.onetcenter.org/database.html

continuous interval [1-7], and our reference job has a score of 4.86.

A job, such as *Information Technology Project Managers*, may have different associated *Tools Used* (e.g., *Desktop computers, Laser printers, Smartphones*, etc.), various *Task Ratings* with scores in the [1-5] range (e.g., *Manage project execution to ensure adherence to budget, schedule, and scope*, with a score value of 4.48). Finally, some examples of *Technology Skills* for our reference occupation include *AJAX* and *Apache Spark*, among others.

The reader can observe that each O*NET job is associated with vectors of 52 Abilities, 33 Knowledge, 35 Skills, and 41 Work Activities. The only variation lies in the score assigned to each value concerning the underlying job. On the other hand, the last three O*NET entities described - *Tools Used*, *Task Ratings*, and *Technology Skills* - only occur when they are required for the underlying job. Additionally, while *Tools Used* and *Technology Skills* do not have any associated scores with an occupation, *Task Ratings* does. In our reference example, there is a requirement for 10 *Tools Used*, 21 *Task Ratings*, and 165 *Technology Skills*, while the same job is consistently associated with four vectors of a fixed size for the first four entities.

The Content Model serves as the conceptual foundation of O*NET, offering a framework that identifies crucial types of information about work and integrates them into a theoretically and empirically sound system. Developed through research on job and organizational analysis, the Content Model embodies a perspective that reflects the character of occupations (via job-oriented descriptors) and people (via worker-oriented descriptors). Moreover, the Content Model facilitates the application of occupational information across jobs, sectors, or industries (cross-occupational descriptors) and within occupations (occupational-specific descriptors). For our knowledge base, we opted to utilize a combination of worker-oriented, job-oriented, and occupation-specific elements. Specifically, information related to job titles, job descriptions, and alternative job titles adheres to the definitions provided by the 2018 SOC System.

IV. ARCHITECTURE

The plug-in architecture is illustrated in Figure 1 within the back-end box. It consists of various blocks, each containing different modules. The Front-end box, depicted in Figure 1, showcases the user interface, which can be implemented by any conversational agent. For simplicity in describing our plug-in architecture, we assume the conversational agent operates within a web app. However, it is crucial to note that the front end is not part of the knowledge plug-in; its representation is solely for a better understanding of how the knowledge plug-in integrates into existing conversational agents and provides a practical demonstration of the plug-in itself.

The User Interface may run through a browser and is presented as a web application communicating with the knowledge plug-in through REST APIs. On the back end, we have the core modules of the proposed knowledge plug-in, which include the Plug-in Interface, Question Comprehension, Response Constructor, and Transformers model blocks. Within the Plug-in Interface, there is the REST API block providing the interface for any server to call back-end functionalities. This design enables decoupling between the knowledge plug-in and any platform utilizing it. The Plug-in Manager within this block orchestrates the processing of received input among other back-end sub-modules.

The Question Comprehension block comprises different modules aiming to identify and collect information for reconstructing the user response. The initial module in Question Comprehension is the NLP module, which analyzes the input text and performs Named Entity Recognition (NER), extracting elements such as nouns, noun phrases, numbers, persons, companies, and nations. Subsequently, the finder module searches for the identified entities in O*NET (referred to as the knowledge base) and forwards the found ones to the grammar module.

The grammar module dynamically constructs a grammar that encompasses all types of requests the plug-in can address, combining the base questions with elements extracted from the input text and retrieved in the knowledge base by the finder module. The grammar module then provides a list of all possible questions. A Transformers model is employed to embed questions and user input, determining the question with the highest similarity to the user's request using the cosine similarity function. The user notices that any specific LLM can be used for sentence similarity, question answering, and summarization within the corresponding blocks of the architecture.

Within the Response Constructor block, the function factory module includes all necessary functions to generate an answer for a specific question type. If the user's request achieves a similarity score higher than an experimentally determined threshold value, the module returns an answer using the specific domain modules, which are the O-NET database and the course recommendation modules. The O-NET database module collects data needed for questions related to jobs, skills, tools, tasks, and work activities. The course recommendation module provides course recommendations that could benefit users in enhancing various aspects of their training.

If the similarity score is not higher than the threshold for both the course recommendation and O*NET database modules, indicating the user is likely asking something else, the open-domain module is triggered. It either searches for the answer over the Internet or redirects the request to the conversational agent incorporating the proposed knowledge plug-in.

The proposed open-domain module retrieves the closest ten results from the Internet, utilizing them as contexts for the question-answering Transformers module. It groups the ten answers by similarity and calculates the sum of the confidence scores of the answers in each group. The open-domain module selects the group with the highest score

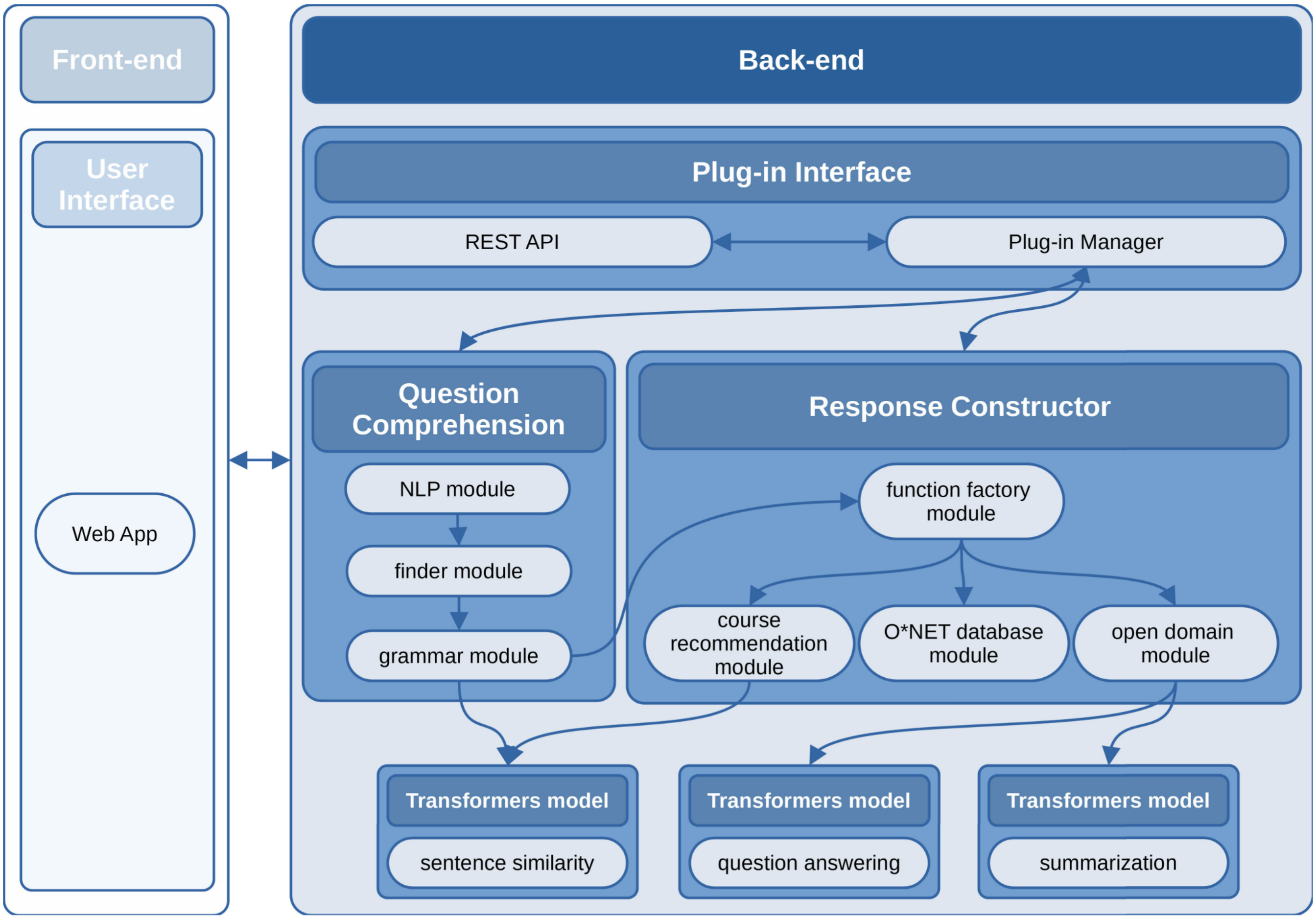


FIGURE 1. Knowledge Plug-in Architecture. The front-end represents any conversational agent and the back end shows the modules of the proposed knowledge plug-in.

to return the answer, which includes the highest-scoring element of the winning group and a concatenated summary of each constituent element. The module only returns the summary if the answer score is lower than a predefined threshold. The open-domain module can be easily disabled when the plug-in is incorporated into a conversational agent with its capabilities to respond to questions in the open domain.

In the following sections, we will provide a detailed explanation of each module contained in both the front-end and the back-end blocks.

A. PLUG IN INTERFACE

The Plug-in Interface operates using Gunicorn,[27] a Python WSGI HTTP Server for UNIX, as a process manager. Gunicorn is based on the pre-fork worker model, where a central main process manages worker processes handling all requests and responses. The system utilizes Unicorn[28] asynchronous workers, compatible with the new ASGI (Asynchronous Server Gateway Interface) standard, ensuring compatibility with the framework used for the REST APIs.

REST APIs rely on FastAPI,[29] a web framework for building APIs with Python, fully compatible with the open standards for APIs (OpenAPI) and JSON Schema. The framework automatically provides API documentation and a graphical user interface for testing purposes. The REST API module receives user input via a GET call and sends it to the Plug-in Manager module for processing.

The results from the Response Constructor, passed through the Plug-in Manager, are returned in JSON format, containing the original user request divided into sentences, lists of nouns, noun phrases, verbs, numbers, entities, dates, any input enclosed in double quotes, elements searched for in the database, those found, the base question from the grammar chosen by the system, the answer, and additional information related to the type of request identified by the system.

Regarding the Plug-in Manager, it receives input text from the REST API module and sequentially queries the NLP and grammar modules of the Question Comprehension block. Specifically, it sends the received text as input to the NLP module and obtains the extracted elements. Then, it sends these items to the grammar module and receives the list

[27] https://gunicorn.org
[28] https://fastapi.tiangolo.com/deployment/server-workers/
[29] https://fastapi.tiangolo.com/

of questions built on them. Once it has collected the list of questions built on the extracted elements, it submits the questions and the user input to the sentence similarity module of the Transformers model to select the question with the highest similarity to the user input.

Finally, it compares the similarity score returned by the Transformers model for that question with an empirically chosen threshold. Then, it calls a different function of the function factory module based on the result of this comparison and the type of the question itself. Each function of the module retrieves the information necessary for one of the recognized question types, constructs the response in natural language, and returns it to the calling module.

B. QUESTION COMPREHENSION

The Question Comprehension block comprises three modules: NLP, finder, and grammar. The NLP module utilizes spaCy,[30] an open-source software library for advanced natural language processing, written in Python and Cython, and published under the MIT license. The ''en_core_web_sm'' model has been employed for text parsing functions. The module takes as input a text and extracts nouns, noun phrases, named entities, and compound expressions in quotes (expressions in quotes can be used by the user to indicate perfect matches, akin to how search engines operate). The reader observes that nouns are generally shorter than noun phrases, which are, in turn, shorter than named entities. For example, in the text *How many jobs require Microsoft Office Professional Plus?*, the identified nouns are *jobs, Microsoft, Office, Professional, Plus*, the identified noun phrases are *How many jobs, Microsoft Office, Professional Plus*, and the only identified named entity is *Microsoft Office Professional Plus*.

From the identified nouns, noun phrases, and named entities, the module eliminates the less important elements, keeping the most complex structures, according to the following logic. If a certain identified noun is also found within noun phrases or named entities or expressions in quotes, then the noun is deleted. If a noun phrase is contained in a certain named entity or expressions in quotes, the noun phrase is removed from the list of identified terms.

Sometimes, nouns or noun phrases can include terms indicating questions such as ''who'', ''what'', or keywords indicating specific actions such as ''list'', ''count'', ''compare'', or indicating the O*NET entities such as ''work activities'', ''abilities'', ''skills'', etc., or others within the domain such as ''jobs'', ''occupations'', or common English articles and pronouns.

For instance, considering the input text *Which occupations have "developing constructive and cooperative working relationships with others, and maintaining them over time" as a Work Activity?*, the identified nouns *occupations, relationships, others, time, Work, Activity* are also found among the noun phrases *which occupations, constructive and cooperative working relationships, others, them, time, Work Activity*, and therefore, they are deleted. The words *which* and *occupations* from the noun phrases are keywords. The noun phrase *Work Activity* is an O*NET entity. The remaining noun phrases are contained in the expression in quotes: *"Developing constructive and cooperative working relationships with others, and maintaining them over time"*. Only this last expression in quotes is, therefore, not deleted.

After performing the deletion of certain elements, we are left with a list of candidate nouns, noun phrases, named entities, and expressions in quotes.

The finder module takes this list of candidate elements as input and searches for each of them within the entities of the O*NET database. For each identified element, it returns the class type to which it belongs (e.g., the noun ''computer programmers'' is identified as a ''job'' in O*NET) and all the other associated information, such as the description and the SOC (Standard Occupational Classification)[31] code and alternate titles.

The semantic search is executed using the sentence similarity model within the Transformers module. This semantic search is computed based on the embeddings of the two underlying elements. To enhance efficiency, the embeddings of all entities in the O*NET database have been pre-calculated and stored. This caching mechanism allows for the computation of the embedding of only the candidate element, resulting in significant savings in computational resources.

For each candidate element, the module retrieves the item in the O*NET database with the highest similarity if the related semantic similarity surpasses an experimentally determined threshold value; otherwise, it is discarded. These items correspond to the O*NET instances and O*NET jobs.

The grammar module aims to generate a list of possible questions starting from a base grammar, intending to extract the question with the highest similarity value to the user input.

TABLE 1. Number of templates for each type of question.

Question	templates
Type 1	3
Type 2	5
Type 3	2
Type 4	6
Type 5	6
Type 6	2
Type 7	6

The candidate questions fall into eight distinct types. Type zero represents a set of ''free'' questions like ''Who are you?'', ''What can you do?'', and so on. Each question of type 0 has one or more fixed answers. Questions of the other types (1-7) are formed using templates designed to serve specific purposes within the STAR project. In these templates, placeholders in curly brackets indicate variables

[30]https://spacy.io/

[31]https://www.bls.gov/soc/2018/home.htm

that can be replaced with the found instances, whereas placeholders in angular brackets are used to expand the grammar with a list of possible values.

The grammar incorporates several templates for each type of base question to maximize the chances of obtaining the most similar question to the user's input. To do so, a collection of potential questions, useful for accessing the data within the O*NET database, was meticulously curated by conducting interviews with academics and industry employees. Following this, tailored queries were crafted to extract the necessary information. This systematic approach involved categorizing the queries into the proposed classifications and outlining corresponding templates. To ensure comprehensive coverage, multiple iterations of each template were formulated. By categorizing the queries and conducting iterations of each template, the system is equipped to handle a wide array of user inputs, match queries accurately, and retrieve relevant data efficiently. In the next section, we will provide further details about this process.

1 CREATION OF THE QUERY TEMPLATES

In this section, we will outline the three steps used to develop the proposed query templates.

- **Identification of O NET tables**: We conducted an extensive study of the O*NET database, focusing on tables that offer the most relevant data for users to explore the unique characteristics of various job types. The tables selected were:
  - *Skills*: Contains 35 skills, with scores for all occupations.
  - *Knowledge*: Contains 33 knowledge areas, with scores for all occupations.
  - *Abilities*: Contains 52 abilities, with scores for all occupations.
  - *Work_activities*: Contains 41 work activities, with scores for all occupations.
  - *Content_model_reference*: Includes names and descriptions of entities in the above tables.
  - *Occupation_data*: Contains titles and descriptions for 1,016 job categories.
  - *Alternate_titles*: Lists alternative titles for the 1,016 job categories.
  - *Technology_skills*: Includes 8,761 different technologies and their associated job categories.
  - *Tools_used*: Details tools associated with the 1,016 job categories.
  - *Task_statements*: Lists 17,975 different tasks and ratings for the 1,016 job categories.
- **Creation of SQL queries based on interviews with professionals**: We engaged in interviews with both academics and industry professionals, presenting the data we selected and soliciting all possible questions they could formulate based on this data. This was done with 2 professors and 4 employees within two companies. From these interactions, we compiled a comprehensive list of unique questions. These were also presented back to the interviewed people to guarantee the correctness and completeness of their formulation. We observed that each question could be answered by extracting data from the selected tables using a limited number of SQL queries. We then crafted tailored SQL queries to extract the necessary information and identified the most common corresponding questions in natural language, which served as the basis for our templates. To encompass all possible variations in the questions, we developed a combinatorial grammar capable of generating all original questions from our unique list when expanded. This grammar was built using a total of 30 base templates.
- **Determining the 7 template types**: We determined seven template types based on the observation that all questions could be answered using seven distinct SQL queries. Consequently, the templates were grouped into these seven categories. The system continuously records all queries and the methods used to answer them. By periodically analyzing these recorded questions, we may identify any that were not adequately addressed due to the absence of a corresponding SQL query. If such gaps are identified, we may seamlessly add new templates to the grammar and corresponding SQL queries to the system, ensuring comprehensive coverage.

In the following, we will provide one example template for each type of base question. In Table 1, the total number of templates for each type of question used in the grammar module is reported.

1) $< list >$ some alternative names for $\{jobs\}$
2) $< list >< entity >$ for $\{jobs\}$
3) Describe $\{jobs\}$
4) How are $\{jobs_0\}$ and $\{jobs_1\}$ similar?
5) $< list >$ the $< jobs >$ which require the $\{\text{name}\}$
6) Count the $< jobs >$ that require the $\{\text{name}\}$
7) Recommend me something about $\{\text{all}\}$

where:

- $< jobs >$ = jobs | occupations | positions
- $< verb >$ = is | are
- $< list >$ = list | what $< verb >$
- $< entity >$ = ability | knowledge | skill | task | work activity | tech-skill | tool

The grammar produces questions of types 1, 2, and 3 when there is at least one job element in the identified instances (O*NET instances plus O*NET jobs), and questions of type 4 when there are two of them. Questions of types 5 and 6 require an O*NET instance. Questions of type 7 are returned if an instance (O*NET instances plus O*NET jobs) has been identified by the finder module. The grammar module is flexible, allowing for extension by adding new types of questions and their related handler functions in the function factory module. Finally, the grammar module returns the list of all the generated questions that it was able to produce to the Plug-in Manager, which will compute the most similar to the user input.

C. RESPONSE CONSTRUCTOR

The Response Constructor block includes four modules: function factory, course recommendation, O*NET database, and open domain.

The function factory module is responsible for constructing an answer to each type of question defined in the grammar module.

- For type-0 questions, if we have a list with several responses associated with the underlying query, the module returns one randomly from the list. Otherwise, it returns the only possible element.
- For questions of type 1-6, the module queries O*NET using the O*NET database module, which contains a set of SQL queries capable of returning the data necessary for the answers. The O*NET database module connects to the database running on MySQL[32] through the "mysql-connector-python" library.[33]
- For type-7 queries, the module queries the course recommendation module.
- For all the queries that do not match any of the types before, the module triggers the open domain module.

One more important task for the function factory module is to generate natural language responses using a predefined template for each question type and data obtained from one of the three other modules in the Response Constructor block. Each question type has its response template, designed to handle correctly the presence of singular or plural items to be returned, especially when associated with numerical quantities or when no results need to be given to the user.

The module constructs responses by merging the structured data from the O*NET database and the course recommendation module with textual models containing the necessary placeholders. The module maintains several textual models suitable for the different possible question types.

As an example, let us assume the user enters: *"I want to know which are the best abilities for actors. Please tell me the first one"*. It is a type 2 question. The function factory model queries the O*NET database module for abilities required for the job "Actors". The database module returns the abilities list for the required job, sorted by importance score. There are 52 abilities (the first 5 are: *"Oral Expression", "Oral Comprehension", "Written Comprehension", "Memorization"*, and *"Speech Clarity"*). Answering the user requires only the first one. The model, therefore, uses the template *"I think it is: {ent}"*, suitable when a single entity is required and substitutes the variable *{ent}* with the name of the first entity in the given list. The resulting answer is therefore: *"I think it is: Oral Expression"*. If the user requested to list two or more abilities the module would use the template *"I think they are: {ent}"*, suitable for a list of entities. If the user asks for a recommendation (e.g., *"Teach me something about Python"*), the function factory module uses the template *"May I suggest a course from {source}, organized by {org}, and entitled: {title}? You can find more information by visiting the related URL: {url}"*. Here it substitutes the variables in curly brackets with data from the course recommendation module. The complete answer, in this case, is *"May I suggest a course from STAR, organized by UDEMY, and entitled Internet of Things with Python Programming? You can find more information by visiting the related URL: https://www.classcentral.com/course/udemy-iot-with-python-23888"*.

The open domain module returns answers already in natural language, to which the function factory concatenates the link of the source web page.

The course recommendation module can suggest courses from multiple sources. Currently, those sources are a list of courses and training assets related to human-centric AI systems in manufacturing environments, selected by STAR project members, and Udemy,[34] which makes use of APIs v2.0 for Udemy Affiliate Users.[35]

This module searches for the most suitable recommendation by calculating the semantic similarity between the course titles and the O*NET instance detected in the input. Then, it returns the course with the highest score, if it is higher than an experimentally determined threshold. It will query the open-domain module if the score is lower than the threshold.

The open domain module takes the user's request as input and, when configured to run in lightweight mode, queries the APIs of the DuckDuckGo search engine[36] using the Python library duckduckgo_search.[37]

In such a setting, it takes the ten most relevant results from the Internet and uses them as contexts for the question-answering model in the Transformers module that we have employed, which returns an answer and a confidence score c for each context-question pair. Then, the open domain module prepares a triangular matrix M where the rows indicate the ten answers ordered in decreasing order of confidence score c returned by the question-answering model. The columns are also the answers in the same order. The element $M[i, j]$ corresponds to the similarity score returned by the sentence similarity Transformer model of the answers i and j. The values of each row which are greater than an experimentally determined threshold are summed. From the row containing the highest sum, we consider the columns whose values are higher than the threshold. Let this list be *candidateanswers*. The final response will contain the answer from *candidateanswers* with the highest score c and the concatenation of the text summary obtained from the summarization model of all the contexts related to the elements in *candidateanswers*.

The module returns only the concatenation of the text summary if the answer score c (in the case of multiple

[32] https://www.mysql.com/
[33] https://pypi.org/project/mysql-connector -python/
[34] https://www.udemy.com/
[35] https://www.udemy.com/developers/affiliate/
[36] https://duckduckgo.com/
[37] https://pypi.org/project/duckduckgo -search/

selected answers, we consider the one with the highest value) is lower than an experimentally determined threshold.

Figure 2 depicts an example of a response to the user's question *"When was the web born?"*. The table on the left shows the answers returned from the question-answering model sorted by confidence scores. The table on the right shows the similarity values (obtained from the sentence similarity model of the Transformers module) of each pair of answers. The row with the highest sum of valid scores is the second and contains the answers n. 2, 3, and 8. Its total score (the sum of confidence scores in the table on the left) is 1.86. In such an example, the three selected answers contain the same text. We consider the highest score of the three which is 0.93, higher than our fixed threshold of 0.8. Therefore, the returned answer by the system would be the concatenation of the selected answer and the summary of the concatenation of the three texts used as input context for the question-answering model seen before: *"1989 - Sir Tim Berners-Lee invented the World Wide Web (WWW) in 1989, while working at CERN. The web was originally conceived and developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world. Sir Tim was born in London and his parents were early computer scientists, working on one of the earliest computers"*. The open domain module strikes a balance between the need for significant computational resources, which may be scarce in certain environments, and the pursuit of high-quality results. However, the reader may notice that the open-domain module can be easily disabled when the plug-in is incorporated into existing conversational agents that are already equipped with their open-domain module.

D. TRANSFORMERS MODELS

As illustrated in the preceding paragraphs, we have implemented three distinct transformer models. One is utilized for sentence similarity to calculate the score between two strings and is employed by the Question Comprehension and Response Constructor modules. Another model is adopted to efficiently handle the question-answering task for questions that trigger the open-domain module, and the third model is used to generate summaries of given contexts. The last two transformer models are employed within the open-domain module to respond to queries that do not match any specific type. More specifically, these models are implemented through the SentenceTransformers[38] [30] framework, which provides a package for easy access to BERT-based models and their derivatives, such as RoBERTa [31], MPNet [32], and ALBERT [33]. These models transform input text into embedding representations. Subsequently, these embeddings can be used to compute similarity, where similar embeddings belong to pieces of text with similar meanings.

To incorporate semantic information into the embedding representations, these models have undergone training using a deep model structure with two branches having the same deep learning layers. By feeding these branches with sentences that are syntactically different but with the same or similar meaning, the model generates embeddings that encapsulate text semantics. These models represent state-of-the-art technology for tasks related to Semantic Textual Similarity (STS) tasks. In the proposed methodology, we utilized the model 'all-MiniLM-L6-v2' for computing text similarities. The goal was to achieve a balance between the model's size (keeping it compact) and its performance.

This model was originally trained using a distillation approach, transferring knowledge from a teacher model to a student model. Subsequently, it underwent fine-tuning on 1 billion sentence pairs using a contrastive learning approach, where the model had to learn the most similar sentence among a set of possible alternatives. It was selected for its small size, speed, and overall good performance across various benchmarks.[39] Similarly, the question-answering and summarization models chosen in our methodology are 'distilbert-base-cased-distilled-squad'[40] and 'sshleifer/distilbart-cnn-12-6'[41] from Hugging Face Transformers. These models offer performances comparable to BERT [34] but are lighter, faster, and require fewer computational resources, making the proposed conversational agent more efficient. They have been incorporated using the abstraction pipeline[42] provided by *Hugging Face*.[43]

E. USER INTERFACE

This section details the user interface developed to integrate our proposed knowledge plug-in with an existing conversational agent, hosted at http://hrilabdemo.ddns.net:8000/bot/. The user interface operates within a web browser and leverages a JavaScript-based web app. User interaction is multimodal, supporting both text and voice inputs. The app utilizes the Web Speech API[44] for speech-to-text and text-to-speech conversions. When the user activates the voice interaction feature, the system awaits the keyword *starbot* to initiate recognition. The microphone icon flashes during this recognition process. Users have the option to customize the activation word, select the language for recognition, and choose the chatbot's voice from those available in the browser. The interface remains functional when minimized while using voice features. User inputs are transmitted to the server hosting our knowledge plug-in through an XMLHttpRequest call. The server responds with a JSON, responding to natural language and additional information such as links to sources or the URL of the suggested course if the course recommendation module is triggered.

[38] https://www.sbert.net/

[39] https://www.sbert.net/docs/pretrained_models.html

[40] https://huggingface.co/distilbert-base-cased-distilled-squad

[41] https://huggingface.co/sshleifer/distilbart-cnn-12-6

[42] https://huggingface.co/docs/transformers/main_classes/pipelines

[43] https://huggingface.co/

[44] https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

User question When was the web born?

Threshold = 0.8

N.	Score	Answer
1	0.95	first 30 years
2	0.93	1989
3	0.67	1989
4	0.65	March 26, 1973
5	0.47	30 April 1993
6	0.46	CERN
7	0.37	more than 50 years ago
8	0.26	1989
9	0.23	How the Web was born History and evolution of the Internet
10	0.11	1993

	1	2	3	4	5	6	7	8	9	10
1	1.00	0.42	0.42	0.30	0.35	0.24	0.53	0.42	0.22	0.29
2		1.00	1.00	0.41	0.62	0.26	0.48	1.00	0.14	0.67
3			1.00	0.41	0.62	0.26	0.48	1.00	0.14	0.67
4				1.00	0.73	0.18	0.41	0.41	0.16	0.53
5					1.00	0.22	0.43	0.62	0.24	0.79
6						1.00	0.21	0.26	0.00	0.29
7							1.00	0.48	0.27	0.44
8								1.00	0.14	0.67
9									1.00	0.18
10										1.00

Winner group N. 2 - total score 0.93 + 0.67 + 0.26 = 1.86

Winner answer N. 2 - score 0.93 > 0.8

Answer returned Answer N. 2 - Summarization of texts from contexts N. 2 + N. 3 + N. 8

FIGURE 2. Open domain answer building example.

V. EVALUATION

The reader may observe that conducting an evaluation against state-of-the-art conversational agents would be inherently unfair. This is due to our proposal being focused solely on a knowledge plug-in as a component for a conversational agent, rather than a comprehensive conversational system.

Current conversational agents lack integrated knowledge plug-ins, leading to non-deterministic responses that may not consistently align with the information present in O*NET. To address this, we conducted a user study to assess the quality and effectiveness of the responses generated and the overall interaction when our knowledge plug-in is integrated into a conversational agent and used by the user.

For this purpose, we organized separate sessions involving 15 employees from three companies within the manufacturing domain, partners of the STAR project [35]. Each session began with a 20-minute introduction to the prototype, outlining its main features. An introduction to the knowledge plug-in was presented to define their tasks. Specifically, users were instructed on the types of tasks the knowledge plug-in is designed for to avoid any expectations of interacting with it as they would with tools like ChatGPT. Subsequently, participants engaged with the prototype for a couple of hours and completed a four-part survey detailing their experience. They created their queries either by following the examples provided for each template in the interface or by writing a query on their own, which would then be automatically matched to one of the defined templates.

The first part provided background information on the interviewed employee. The subsequent part featured a standard System Usability Scale (SUS)[45] [36] questionnaire aimed at assessing the overall usability of the proposed system. The third section comprised questions related to the quality of interaction, the provided recommendations, and information, using a [1-5] scale for responses. The concluding part consisted of five open-ended questions, prompting the employee to share insights on the system's strengths, weaknesses, and suggestions for additional features. The detailed results are provided in the subsequent paragraphs.

A. USER BACKGROUND

Regarding the user demographic, we chose fifteen individuals from diverse backgrounds, encompassing three different companies.

On average, these participants held 9.2 years of professional experience. Their areas of expertise spanned a wide range, including software engineering, robotics, IoT, software development, social robotics, web and desktop applications development, manufacturing, digitalization, machine learning, building performance, energy efficiency, renewable heating and cooling, and wastewater.

B. SUS QUESTIONNAIRE

The score we derived from the SUS questionnaire is 80.7/100, corresponding to an A grade and placing our prototype in the 90th percentile rank.[46]

The users' responses to the SUS questionnaire are illustrated in Figures 3 and 4. It is noteworthy that while the even questions (Figure 4) are framed negatively (lower scores indicating better results), the odd questions (Figure 3) are framed positively (higher scores indicating better results).

[45] System Usability Scale (SUS) - https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

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

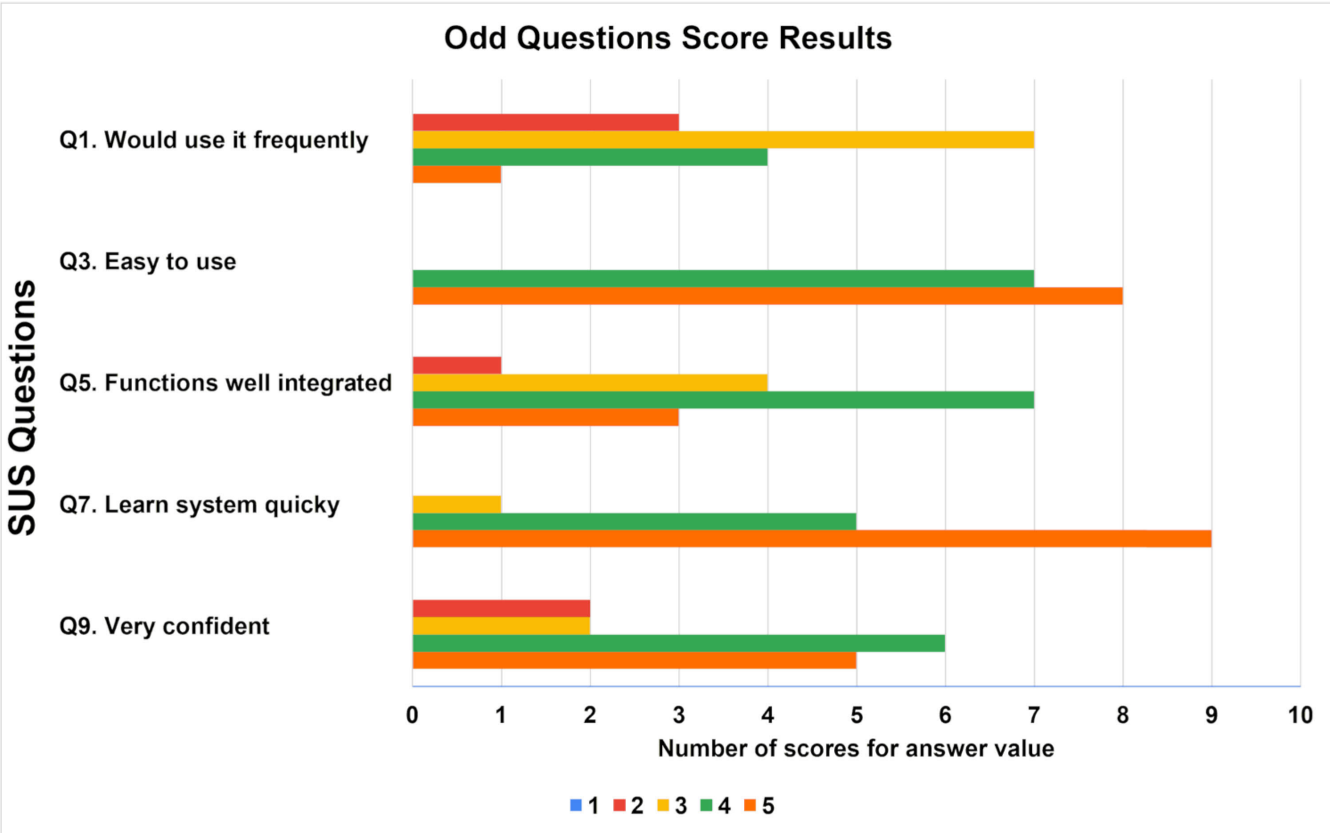


FIGURE 3. SUS Questionnaire results Positive Questions).

In the subsequent sections, we will present the averages and standard deviations for selected questionnaire items. Respondents expressed a high likelihood of frequent system use, indicated by an average score of 3 20 0 83. They found the system to be notably uncomplicated, with an average score of 1 40 0 49. Participants perceived the system as easy to use, recording an average score of 4 53 0 50, and they felt confident in navigating it independently, without requiring technical assistance (average score of 1 53 1 09).

Furthermore, respondents acknowledged the well-integrated functionalities of the system, resulting in a high average score of 3 80 0 83 and perceived consistency (average score of 2 20 0 98). The participants anticipated a quick learning curve for the system, reflected in an average score of 4 53 0 62, and appreciated its non-cumbersome nature (average score of 1 47 0 50). Lastly, users reported feeling very confident using the system, as evidenced by an average score of 3 93 1 00, and expressed that it did not necessitate extensive learning before becoming familiar with it (average score of 1 13 0 34).

To check if there is a difference in the mean we applied a one-way ANOVA test at 0.05 significant level to analyze the data from three groups, considering three different profiles of users (Innovation Management and Business Development, ICT, and Engineering, Research, and Academia). Each of the three groups included five users. We followed the ANOVA methodology as outlined by Cuemath[47] and used the F-table to compute the ANOVA test at a 0.05 significance level for the SUS questionnaire (each of the 10 questions). The results indicate that the null hypothesis is not rejected, showing that there is no significant difference in the mean scores among the groups. This finding suggests that the different profiles of users shared similar advantages and concerns about the benefits and limitations of the proposed tool. We also conducted an ANOVA test for the SUS questionnaire (each of the 10 questions) using a different classification of the users, where each company was treated as a separate group. This resulted in three groups with varying numbers of users per group. Even in this scenario, the null hypothesis was not rejected, indicating no significant difference in the mean scores among the groups. This finding supports the notion that employees from the three companies exhibited similar behaviors and responses. This consistency is likely due to the company operating in the same domain and having similar requirements for the proposed approach. The results of the ANOVA test can be found at http://192.167.149.18/SUS_groups_results.xlsx.

C. QUALITY ASSESSMENT

The evaluators were tasked with assessing the quality of interaction with the developed knowledge plug-in. Three

[47]https://www.cuemath.com/anova-formula/

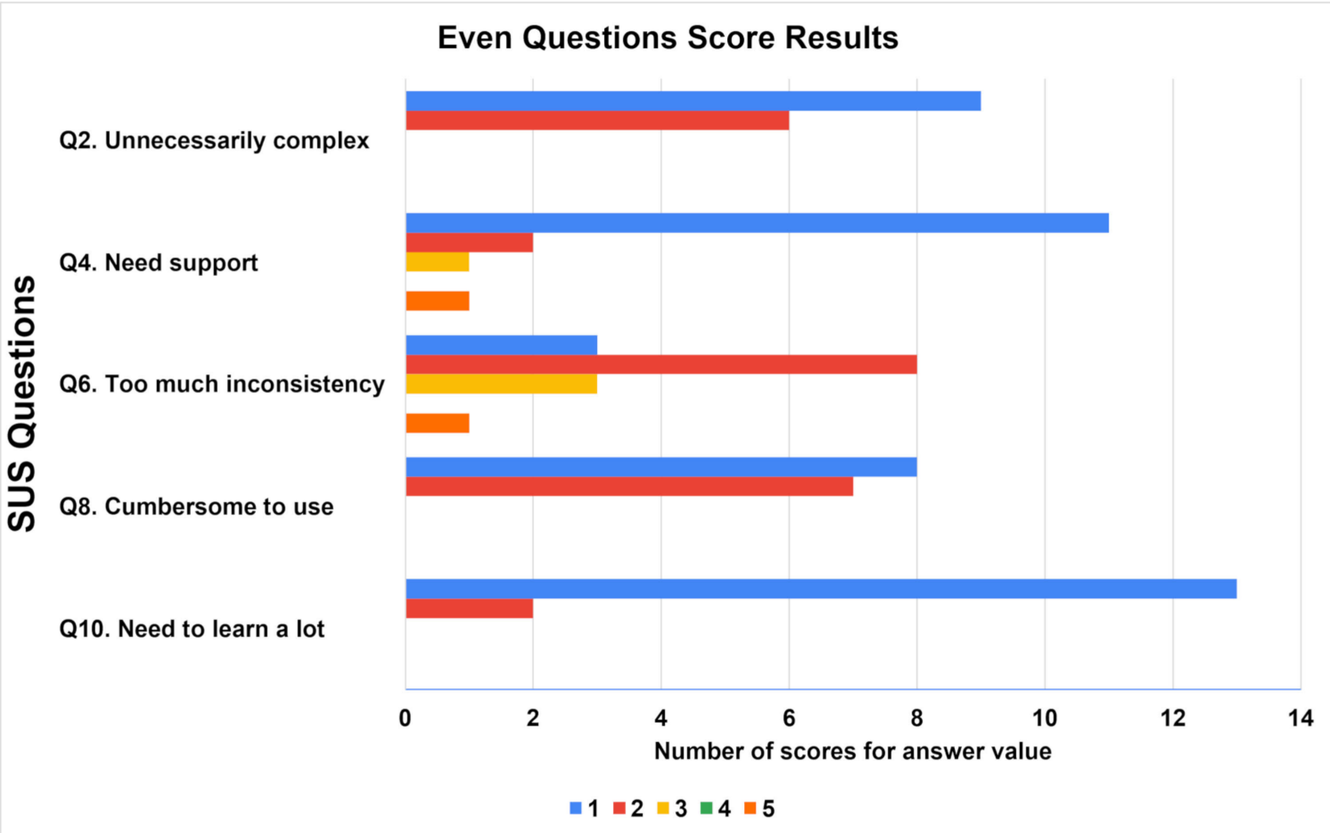


FIGURE 4. SUS Questionnaire results Negative Questions).

questions were presented to them, each scored on a scale from 1 to 5 (where 1 indicated low quality, and 5 indicated high quality). These questions were:

- *How do you assess the quality of the interaction with the STAR-Bot interface?*
- *How do you rate the recommendations provided by STAR-Bot?*
- *How do you rate the provision of information from STAR-bot about a certain job?*

TABLE 2. Quality assessment questions.

Question	Average	Standard deviation
How do you assess the quality of the interaction with the STAR-Bot interface?	3.87	±0.81
How do you rate the recommendations provided by STAR-Bot?	3.87	±0.81
How do you rate the provision of information from STAR-bot about a certain job?	3.67	±0.87

The average and standard deviation of their responses are detailed in Table 2.

Additionally, concerning each of the three aforementioned questions, annotators were allowed to provide feedback. This feedback pertained to the quality of interaction with the plug-in, the recommendations offered, and the information provided for each job.

One user expressed dissatisfaction with the slowness of the interaction, despite its accuracy. Another suggested improving the system's responses to enhance naturalness. A different user proposed the inclusion of a feature allowing the bot to remember previous conversations with the user.

Regarding recommendations, users were generally content with the wide range provided across various fields. However, one user desired the system to offer a learning curve associated with each recommendation.

For the provision of information, a user suggested including details about average salaries.

D. OPEN QUESTIONS

This section will discuss the responses provided by users to the open-ended questions. It is important to note that not all fifteen users responded to every question.

Q1. What are the main strengths of STAR-Bot?

Nine users found it to be well-designed, elegant, and featuring a clear user interface. Another user appreciated the capability to provide specific and up-to-date information about jobs and skills. Additionally, one user valued the links provided by the system to external sources, while another user commended its flexibility.

Q2. What are the main weaknesses of STAR-Bot? One user noted that there might be no information available for specific jobs. Another user expressed dissatisfaction with the length of the bot window when using Microsoft Edge as a browser. Furthermore, the generated list of similarities (especially when comparing a couple of jobs) was considered challenging to read in the current format. Additionally, a concern was raised about the wait time for a question and the absence of feedback on whether the question had been accepted.

Another user desired more advanced conversational capabilities, such as the ability to remember user responses and adjust behavior accordingly. Moreover, two users expressed a preference for a method to handle potential ambiguity in questions. Notably, voice recognition did not function for a user utilizing Microsoft Edge as a browser.

Q3. Can you think of any additional features to be included in STAR-Bot? Users offered various suggestions for improvement. For instance, one suggested incorporating a job skills/abilities self-assessment and expanding training recommendations. Another user expressed a desire to augment the knowledge base by including real job offers with salary information and enhancing job recommendations. Related to this, there was a suggestion to incorporate job statistics in a specified domain. Additionally, a requested feature was the implementation of a history feature to track all user questions.

Q4. Can you think of any additional types of queries for STAR-Bot?

One user expressed a desire for the system to respond to questions such as, *I am a Computer Scientist, what are the usual career changes?"*. Another user suggested the inclusion of queries related to the differences between two specified jobs. While the system currently provides similarities between two jobs, having a list of differences would offer additional valuable information. Another suggestion focused on job interviews, with an example query like, *How should I prepare for an interview for a computer science position?"*.

Q5. What would you add to increase the accuracy/comprehensiveness of the information returned by STAR-Bot? The users provided the following suggestions: 1) include more recommendation sources such as whitepapers and academic papers; 2) remove the title of the article that the bot has found to enhance the naturalness and comprehensiveness of the information; 3) introduce a clear tag at the beginning of the answer whenever the system searches online, specifying the source (e.g., Google or similar); 4) offer more than one link to web resources related to the bot's answer; 5) when applicable, provide responses using tabular views.

VI. POST EVALUATION ANALYSIS

The proposed knowledge plug-in seamlessly integrates into a web page, offering assistance to users with inquiries about the labor domain. In this section, we outline the suggestions received for a web page incorporating a minimal conversational agent utilizing our knowledge plug-in, based on the feedback received during user evaluations. Additionally, we describe the modifications made to our proof of concept in response to this feedback.

Firstly, we introduced animations to enhance the user experience while the system retrieves data, providing clarity during periods of waiting. Another visual improvement includes the automatic adaptation of the bot window to fit the current screen size.

Concerning the provided information, we initially offered external resource links exclusively for suggested courses. Now, references to the source pages of the information retrieved from internet searches have also been incorporated into the responses. Furthermore, a filtering feature has been added to discard irrelevant information from internet content that does not pertain to the job market domain.

The architecture has been extended to facilitate the integration of recommendation material from various sources, including scientific articles.

Future developments currently in progress encompass:

- Enabling multiple interactions on the same query, allowing users to delve deeper into the requested topic.
- Incorporating a system for saving or exporting answers deemed most interesting by the user.
- Implementing the capability for the agent to answer questions about its operation and the range of queries it can address.
- Enhancing the system to allow further interactions in cases where the user's request is ambiguous.
- Enabling the agent to assist users in performing job skills/abilities self-assessment.
- Integrating with GPT family models, triggering our system only when users' questions are related to the job market domain.

It is essential to reiterate that our objective is not to propose a new conversational agent for the open domain. Instead, we offer a knowledge plug-in designed for existing conversational agents to precisely and effectively respond to questions within the labor domain, aligning with the knowledge in O*NET. When integrated into existing conversational agents, the open domain module should be configured to redirect general questions to the underlying conversational agents. Additionally, our architecture is versatile and can be reused to create other knowledge plug-ins in different domains.

VII. SCENARIOS

In this section, we present two potential usage scenarios for the proposed tool, aiming to showcase the impact and efficiency it may have within the labor market domain. The reliance of our system on a resource like O*NET enhances its suitability and trustworthiness for users.

The first scenario involves an unemployed individual searching for a job in New York. The second scenario features a web programmer aspiring to transition from their current job to a research career.

Next, we describe an interaction between a user and our tool related to scenario 1.

User: What are the most requested jobs in New York?
Bot: Stacker - The 10 largest occupations in America employ 30.5 million workers, representing 21% of all workers. Office workers, package handlers, food prep workers, and truck drivers are among the most common jobs in New York City. Stacker - Most Common Jobs in NYC.
User: Describe package handlers job
Bot: Manually move freight, stock, luggage, or other materials, or perform other general labor. Includes all manual laborers not elsewhere classified.
User: Describe food prep workers job
Bot: Perform a variety of food preparation duties other than cooking, such as preparing cold foods and shellfish, slicing meat, and brewing coffee or tea.
User: Other names for food prep workers?
Bot: Some alternative names for Food Preparation Workers are: Cafeteria Aide, Carver, Caterer, Caterer Helper, Catering Assistant, Chicken and Fish Butcher, Coffee Brewer, Coffee Maker, Cold Meat Cook, Crew Mess Attendant
User: Tell me common tasks for food prep workers
Bot: I think they are: Clean and sanitize work areas, equipment, utensils, dishes, or silverware. - Operate cash register, handle money, and give correct change. - Store food in designated containers and storage areas to prevent spoilage. - Portion and wrap the food, or place it directly on plates for service to patrons. - Take and record temperature of food and food storage areas, such as refrigerators and freezers.
User: What knowledge does a Food Preparation Worker need?
Bot: I think they are: Customer and Personal Service - Administration and Management - Public Safety and Security - Food Production - English Language
User: Recommend me something about Customer Service
Bot: I may suggest a course from Udemy, organized by Udemy, and entitled: Customer Service, Customer Support, And Customer Experience. You can find more information by visiting the related url[48]
User: What skills must a Food Preparation Worker have?
Bot: I think they are: Active Listening - Speaking - Social Perceptiveness - Coordination - Service Orientation
User: Give me something about Social Perceptiveness
Bot: May I suggest a course from Udemy, organized by Udemy, and entitled: The Secrets of Confidence and Communication. You can find more information by visiting the related URL: https://www.udemy.com/course/confidence-and-communication/.
User: Thanks
Bot: You are welcome
User: Bye
Bot: See you later

In the subsequent section, we provide another example of an interaction between a user and our system related to scenario 2.

User: describe researchers job
Bot: Conduct research in fundamental mathematics or in application of
mathematical techniques to science, management, and other fields. Solve problems in various fields using
mathematical methods.
User: List the most important tasks for researchers
Bot: I think they are: Address the relationships of quantities, magnitudes, and forms through the use of numbers and symbols. - Disseminate research by writing reports, publishing papers, or presenting at professional conferences. - Maintain knowledge in the field by reading professional journals, talking with other mathematicians, and attending professional conferences. - Apply mathematical theories and techniques to the solution of practical problems in business, engineering, the sciences, or other fields. - Develop mathematical or statistical models of phenomena to be used for analysis or for computational simulation.
User: tell me the most important knowledge for researchers
Bot: I think they are: Mathematics - English Language - Computers and Electronics - Education and Training - Engineering and Technology
User: I need to improve on maths
Bot: May I suggest a course from Udemy, organized by Udemy, and entitled: Discrete Mathematics. You can find more information by visiting the related url: https://www.udemy.com/course/discrete-math/
User: compare programmers to researchers
Bot: Programmer and Researcher have: 1 (14.0%) common tool (Desktop computers), 33 (11.0%) common tech-skills (first 5 items in alphabetical order: Microsoft Access - Minitab - Perl - Python - Tableau), and 0 (0.0%) common task. They

[48]https://www.udemy.com/course/how-to-find-your-voice-the-resonates-with-your-customers/

also have: 94.0% ability similarity (first 5 items sorted by least difference: Written Comprehension - Problem Sensitivity - Flexibility of Closure - Spatial Orientation - Multilimb Coordination), 88.0% knowledge similarity (first 5 items sorted by least difference: Therapy and Counseling - Building and Construction - Food Production - Medicine and Dentistry - Mechanical), 89.0% skill similarity (first 5 items sorted by least difference: Monitoring - Persuasion - Equipment Selection - Equipment Maintenance - Repairing), and 92.0% work-activity similarity (first 5 items sorted by least difference: Scheduling Work and Activities - Processing Information - Getting Information - Resolving Conflicts and Negotiating with Others - Judging the Qualities of Objects, Services, or People). 91.0% total similarity for scored entities (abilities, knowledge, skills, work activities) and 55.0% global similarity score. The elements that most characterize the first job (Programmer) are: Working with Computers - Computers and Electronics - Programming - Processing Information - Analyzing Data or Information. The elements that most characterize the second job (Researcher) are: Mathematical Reasoning - Mathematics - Mathematics - Analyzing Data or Information - Making Decisions and Solving Problems.

User: I need something about Making Decisions and Solving Problems

Bot: May I suggest a course from Udemy, organized by Udemy, and entitled: Problem Solving and Decision Making Creatively. You can find more information by visiting the related url.[49]

User: I need also something about Analyzing Data and Information

Bot: May I suggest a course from STAR, organized by EdX, and entitled: Data Science for Executives. You can find more information by visiting the related URL: https://www.edx.org/professional- [...].

VIII. CONCLUSION

The labor market is undergoing rapid transformations, impacting both job offers and job seekers. Certain positions face an excess of job offers that struggle to attract suitable candidates. This challenge is particularly evident in roles that require expertise from various domains, where candidates might dismiss opportunities without realizing that acquiring or refining a skill could make them an ideal fit. Not only is the labor market evolving swiftly, but training opportunities are also dynamic, providing candidates with diverse possibilities to acquire new skills through study or practice. The primary challenge lies in efficiently and consistently accessing this information.

To address these dynamic demands with reliable and pertinent information, this article introduces a knowledge plug-in for existing conversational agents. This plug-in is designed to respond to specific queries related to the job market domain.

To support our proposal, we have defined an architecture for a knowledge plug-in that incorporates information from the O*NET occupational database and a recommendation engine for training materials. These components can be seamlessly integrated into existing conversational agents. Additionally, we propose an innovative approach that leverages Deep Learning and NLP techniques to dynamically generate different grammars. These grammars are employed to semantically associate user questions with a predefined list, enhancing the user's interaction. Furthermore, we present an innovative scheme to enhance the results of the question-answering task by matching responses against a taxonomy.

When integrated into an existing conversational agent, disabling the open domain module in the proposed architecture allows the underlying conversational agent to utilize its existing capabilities.

Our architecture revolves around a back-end that includes Question Comprehension and Response Constructor modules, both backed by Transformers model blocks. The former, supported by an NLP module and a grammar management module, is responsible for identifying and gathering information. This information is later processed to interact with the Transformers models. The latter is responsible for reconstructing user responses and accessing modules related to the O*NET database. Additionally, it incorporates a module for open domain generation in cases where the question is off-topic, and the answer cannot be found in the occupational database.

While ChatGPT has demonstrated impressive conversational capabilities across various domains, it may produce less accurate and potentially misleading answers in specific domains. For instance, it might generate dissimilar responses when asked about the main tasks associated with a particular position. To ensure deterministic and accurate information aligned with a widely recognized resource and to mitigate potential hallucination issues associated with existing conversational agents, we have chosen to focus on responses related to skills and knowledge using the O*NET database. This database serves as a comprehensive repository of multiple variables associated with various occupations.

We implemented an online chatbot based on the insights and recommendations provided in this article and conducted a user evaluation, including the calculation of the SUS score, to identify areas for improvement in future iterations. The achieved SUS score of over 80.7% reflects very positive user feedback, supporting the effectiveness of the approach outlined in this article. Particularly noteworthy is the chatbot's ability to complement other general-purpose artificial intelligence chatbots. While existing agents can

[49]https://www.udemy.com/course/problem-solving-and-decision-making-creatively/

handle general information, our engine excels in delivering specific, on-topic information. This strength stems from a robust architecture and efficient conversation management, leveraging reliable sources to provide users with valid and relevant information. Additionally, our engine can seamlessly integrate with existing conversational agents, harnessing their capabilities in the open domain.

Although our user study yielded satisfying results, a few limitations indicate areas for further improvement. First, the study revealed that additional research and enhancement should be performed on user engagement and perceived utility. Second, while the current implementation of the knowledge plug-in has been validated using the O*NET resource, its applicability with other resources remains unexplored and warrants further investigation.

Addressing user feedback, one prominent challenge is enhancing the knowledge plug-in's ability to remember conversations, a feature often suggested during the validation process. This includes incorporating memory and recall capabilities, features already present in several chatbots utilizing supervised and reinforcement learning techniques.

Our future work will focus on these aspects, alongside exploring new applications like virtual interviewers. We also plan to augment current databases with information from well-established occupational databases such as ESCO, the multilingual database for the classification of European Skills, Competences, and Occupations published by the European Commission, or the International Standard Classification of Occupations (ISCO). Additionally, we envision creating a semantic representation, such as knowledge graphs, for these occupational resources. This will enable their linkage to external knowledge bases, enhancing content and introducing new functionalities for chatbots, such as job offer searches.

Furthermore, we aim to investigate how complex user queries can be broken down into simpler sub-questions. This approach ensures a broader range of user requests can be answered using our proposed grammar, reducing reliance on the open-domain module. This not only provides higher control but also ensures the provenance of answers, facilitating the exploration of issues over time.

Lastly, a potential future direction involves integrating RAG with our approach. This would allow us to incorporate LLMs while limiting the hallucination issue within our architecture. Essentially, our method can be utilized to establish context by searching the database for pertinent information extracted from the user's query. Subsequently, RAG could be applied to generate a dynamic answer based on the identified context. Although the output of RAG may still be vulnerable to hallucinations and incomplete answers, evaluating the strategy for defining context can be explored as part of future work.

In summary, our work addresses various needs in the continually evolving job market, paving the way for the development of diverse functionalities to assist both candidates and employers in filling job positions.

REFERENCES

[1] M. Krause and T. A. Lubik. (2007). *On-the-Job Search and the Cyclical Dynamics of the Labor Market*. [Online]. Available: https://ideas.repec.org/p/zbw/bubdp1/6136.html

[2] O. H. Hamid, N. L. Smith, and A. Barzanji, "Automation, per se, is not job elimination: How artificial intelligence forwards cooperative human-machine coexistence," in *Proc. IEEE 15th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2017, pp. 899–904.

[3] C. Li, Y. Zhang, X. Niu, F. Chen, and H. Zhou, "Does artificial intelligence promote or inhibit on-the-job learning? Human reactions to AI at work," *Systems*, vol. 11, no. 3, p. 114, Feb. 2023.

[4] S. Mohamad Suhaili, N. Salim, and M. N. Jambli, "Service chatbots: A systematic review," *Expert Syst. Appl.*, vol. 184, Dec. 2021, Art. no. 115461.

[5] A. Meloni, S. Angioni, A. Salatino, F. Osborne, D. Reforgiato Recupero, and E. Motta, "Integrating conversational agents and knowledge graphs within the scholarly domain," *IEEE Access*, vol. 11, pp. 22468–22489, 2023.

[6] R. Alonso, D. Dessí, A. Meloni, and D. R. Recupero, "A novel approach for job matching and skill recommendation using transformers and the o*net database," *Big Data Res.*, vol. 1, pp. 1–13, Sep. 2023.

[7] R. Alonso, D. Dessì, A. Meloni, and D. Reforgiato Recupero, "A general and NLP-based architecture to perform recommendation: A use case for online job search and skills acquisition," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, Mar. 2023, pp. 1–20.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 1–23.

[9] J. Dhameliya and N. Desai, "Job recommender systems: A survey," in *Proc. Innov. Power Adv. Comput. Technol.*, vol. 1, Mar. 2019, pp. 1–5.

[10] J. Jarrett and M. B. Blake, "Using collaborative filtering to automate worker-job recommendations for crowdsourcing services," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2016, pp. 641–645.

[11] W. Shalaby, B. AlAila, M. Korayem, L. Pournajaf, K. AlJadda, S. Quinn, and W. Zadrozny, "Help me find a job: A graph-based approach for job recommendation at scale," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2017, pp. 1544–1553.

[12] A. Giabelli, L. Malandri, F. Mercorio, M. Mezzanzanica, and A. Seveso, "Skills2Job: A recommender system that encodes job offer embeddings on graph databases," *Appl. Soft Comput.*, vol. 101, Mar. 2021, Art. no. 107049.

[13] Y. Lu, S. El Helou, and D. Gillet, "A recommender system for job seeking and recruiting website," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 963–966.

[14] R. Mishra and S. Rathi, "Enhanced DSSM (deep semantic structure modelling) technique for job recommendation," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7790–7802, Oct. 2022.

[15] C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding, and P. Li, "Person-job fit: Adapting the right talent for the right job with joint representation learning," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 3, pp. 1–17, Sep. 2018.

[16] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 25–34.

[17] R. Ramanath, H. Inan, G. Polatkan, B. Hu, Q. Guo, C. Ozcaglar, X. Wu, K. Kenthapadi, and S. C. Geyik, "Towards deep and representation learning for talent search at LinkedIn," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 2253–2261.

[18] E. U. Reshma and P. C. Remya, "A review of different approaches in natural language interfaces to databases," in *Proc. Int. Conf. Intell. Sustain. Syst.*, Dec. 2017, pp. 801–804.

[19] C. Baik, H. V. Jagadish, and Y. Li, "Bridging the semantic gap with SQL query logs in natural language interfaces to databases," in *Proc. IEEE 35th Int. Conf. Data Eng.*, Apr. 2019, pp. 374–385.

[20] S. Abbas, M. U. Khan, S. U. Lee, A. Abbas, and A. K. Bashir, "A review of NLIDB with deep learning: Findings, challenges and open issues," *IEEE Access*, vol. 10, pp. 14927–14945, 2022.

[21] X. Victoria Lin, R. Socher, and C. Xiong, "Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing," 2020, *arXiv:2012.12627*.

[22] T. Scholak, N. Schucher, and D. Bahdanau, "PICARD: Parsing incrementally for constrained auto-regressive decoding from language models," 2021, *arXiv:2109.05093*.

[23] F. Pecune, S. Murali, V. Tsai, Y. Matsuyama, and J. Cassell, "A model of social explanations for a conversational movie recommendation system," in *Proc. 7th Int. Conf. Hum.-Agent Interact.*, Sep. 2019, pp. 135–143.

[24] A. Fadhil and S. Gabrielli, "Addressing challenges in promoting healthy lifestyles: The al-chatbot approach," in *Proc. 11th EAI Int. Conf. Pervasive Comput. Technol. Healthcare*, May 2017, pp. 261–265.

[25] M. Lee, S. Ackermans, N. van As, H. Chang, E. Lucas, and W. Ijsselsteijn, "Caring for vincent: A chatbot for self-compassion," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–13.

[26] 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.*, May 2017, pp. 3506–3510.

[27] T. Lee, T. Zhu, S. Liu, L. Trac, Z. Huang, and Y. Chen, "CASExplorer: A conversational academic and career advisor for college students," in *Proc. 9th Int. Symp. Chin. CHI*, Oct. 2021, pp. 112–116.

[28] M. X. Zhou, G. Mark, J. Li, and H. Yang, "Trusting virtual agents: The effect of personality," *ACM Trans. Interact. Intell. Syst.*, vol. 9, nos. 2–3, pp. 1–36, Sep. 2019.

[29] G. M. Biancofiore, T. Di Noia, E. Di Sciascio, F. Narducci, and P. Pastore, "Guapp: A knowledge-aware conversational agent for job recommendation," in *Proc. Joint KaRS ComplexRec Workshop. CEUR-WS*, 2021, pp. 1–17.

[30] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," 2019, *arXiv:1908.10084*.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[32] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and permuted pre-training for language understanding," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2020, pp. 16857–16867.

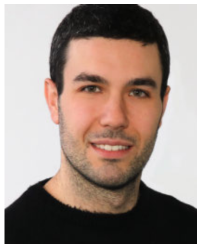
[33] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[35] R. Alroobaea and P. J. Mayhew, "How many participants are really enough for usability studies?" in *Proc. Sci. Inf. Conf.*, Aug. 2014, pp. 48–56.

[36] J. Brooke, "SUS: A 'quick and dirty' usability scale," *Usability Eval. Ind.*, vol. 189, pp. 1–27, Sep. 1996.

RUBÉN ALONSO received the Diploma degree in advanced studies on computing systems from the University of the Basque Country and the degree in computer science and engineering from the University of Deusto. He has been involved in industrial and research projects, since 2004, mainly related to computational trust, security, and embedded systems. He was part of the Trust and Security Unit, European Software Institute, and was responsible for the Security and Embedded Systems Group of Visual Tools. He was also a Visiting Researcher with the VTT Technical Research Centre of Finland and the University of Trento. Currently, he coordinates the ICT and Robotics Department of R2M Solution. His research interests include Linux embedded systems, robotics, IT security, and data mashups. He has experience in the European Research Programmes and industrial experience with various firms.



DANILO DESSÍ received the master's and Ph.D. degrees from the University of Cagliari, Italy. He has been a Senior Researcher with the KTS Department, GESIS Leibniz Institute for the Social Sciences, since October 2022. Previously, he was a Researcher with the University of Cagliari and a Postdoctoral Researcher/Senior Researcher with the FIZ Karlsruhe Leibniz Institute for Information Infrastructure and Karlsruhe Institute of Technology (KIT). He has been a Visiting Researcher in the following centers all around the world: Philips Research, Eindhoven, in 2016; Center for Data Science NYU, New York City, in 2017; Knowledge Media Institute, The Open University, Milton Keynes, in 2018; and University of Paris 13, Paris, in 2019. He is the co-author of CS-KG, a large-scale automatically generated knowledge graph about computer science and a developer of its construction pipeline. He has co-authored more than 40 research publications in top-tier international conferences and journals. His current research interests include artificial intelligence, knowledge graphs, science of science, deep learning, and natural language processing.

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. His research interests include natural language processing, human–robot interaction, conversational agents, and semantic web. He is the Key Developer of amr2fred, a tool for translating abstract meaning representation to motif-based linguistic knowledge graphs and a Key Developer of AIDA-bot, a conversational agent to explore scholarly knowledge graphs. Since 1998, he has been working for local public administrations as a System Administrator, an Expert in geographic information systems and CAD, a Software Developer, and an IT Infrastructure Designer.

DIEGO REFORGIATO RECUPERO received the Ph.D. degree in computer science from the University of Naples Federico II, Italy, in 2004. He has been a Full Professor with the Department of Mathematics and Computer Science, University of Cagliari, Italy, since February 2022. From 2005 to 2008, he was a Postdoctoral Researcher with the University of Maryland College Park, USA. He won different awards in his career (such as the Marie Curie International Reintegration Grant, Marie Curie Innovative Training Network, Best Researcher Award from the University of Catania, Computer World Horizon Award, Telecom Working Capital, Startup Weekend, and Best Paper Award). 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 grids. He is the author of more than 200 conference and journal articles in these research fields, with more than 2800 citations.

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