An Intelligent Multi-Agent System using XML for Adaptive Employment Agency Management

VINCENZO BARRILE[1,*], PIERO FRANCESCO SPANO'[1], EMANUELA GENOVESE[1], GABRIELE BARRILE[2], GIUSEPPE MARIA MEDURI[1]
[1]Department of Civil Engineering, Energy, Environment and Materials (DICEAM),
Mediterranea University of Reggio Calabria,
Via Zehender (loc. Feo di Vito), 89124, Reggio Calabria (RC),
ITALY

[2]LUISS - Libera Università Internazionale degli Studi Sociali Guido Carli,
Viale Pola 12, 00198 Roma,
ITALY

**Corresponding Author*

Abstract: - In recent years, e-commerce has become a significant social and cultural phenomenon. Many organizations now offer their services online. Within this context, online recruitment services play an important role in supporting individuals in job searches and assisting companies in finding personnel. Conversely, the potential that AI forecasting systems offer in distinct application fields is well known using increasingly high-performance algorithms that are particularly promising and certainly used successfully in job searches. In this field, in general, companies insert their job offers (job proposals) into a database; individuals are supported in their search for a job offer using a search engine based on classic Information Retrieval (IR) techniques. Regarding this, it is presumable that the Information Retrieval techniques currently used by recruitment services, which do not involve the use of rich user profiles, can provide a single individual with a large number of job offers, many of which are of little interest for him. This result could cause strong dissatisfaction for the user, and his renunciation of the use of such services. The Geomatics Laboratory, in the context of forecasting studies in the territorial and environmental fields, is experimenting with forecasting systems also in the business and economics fields to create a customized search engine, proposing a Recommender System developed using intelligent agents and based on XML. This system leverages detailed user profiles to assist users in personalized job offer searches, which combines classic Information Retrieval techniques with User Modeling techniques and artificial intelligence techniques. As a result, in generic domains, our system quickly achieves satisfactory results; however, it struggles to maintain this performance after many sessions. Conversely, in specialized domains, while our system requires many sessions to reach satisfactory performance initially, it consistently delivers outstanding performance once this phase is complete.

Key-Words: - Recommender-System, Information-Retrieval, User-Modeling, Multi-Agent, XML, Artificial Intelligence, Forecasting Systems.

Received: March 5, 2024. Revised: August 2, 2024. Accepted: August 27, 2024. Published: September 30, 2024.

1 Introduction

Some approaches based on Recommender Systems have been proposed in the scientific literature to build personalized search engines capable of acting in different application contexts, such as web browsing, e-commerce, e-learning, and so on, [1], [2], [3], [4], [5], [6], [7]. Furthermore, in some of these proposals, Recommender Systems have been combined with intelligent agent technology to make them autonomous, reactive, and proactive. The combination of artificial intelligence (AI) technologies with organizational practices has seen a significant rise, encompassing various domains from job recruitment to social network interactions. The rapid advancement and adoption of AI, particularly generative AI tools, necessitate a comprehensive understanding of their implications on workplace dynamics, employee experiences, and job designs. While some view AI as a catalyst for enhancing efficiency and productivity, others express concerns regarding its impact on human

workers. In this field, it is important to focus on three different areas of interest: the impact that artificial intelligence is having today on job search and job experience, the role of AI within social networks to search for work, and the integration of this technology within staff recruitment processes.
The use of AI in organizations has been studied on multiple levels, highlighting issues such as collaboration between humans and artificial intelligence, the perception of the capabilities of algorithms compared to human ones, the attitude of workers towards artificial intelligence, the use of AI as a control tool, and the broader market implications of the use of AI in the workplace. These themes highlight the complex interaction between AI and humans, offering insights into both the potential benefits, advantages, and challenges of integrating AI into various level contexts.

In the last years, we assisted to the emergence of social networks as tools for employment and business-related user interactions leading to a significant influence of artificial intelligence in this field. As users increasingly depend on instant messaging applications, and specialized social networks for job searching, recommendation systems based on AI have become fundamental. Evaluations of such systems demonstrate high acceptance rates, underscoring the effectiveness of AI in managing large volumes of content and facilitating user engagement, [8], [9]. Moreover, the adoption of AI in recruitment processes has revolutionized talent acquisition, reducing complexities in candidate sourcing, screening, and evaluation. AI technologies not only streamline administrative tasks but also promise to improve hiring quality and mitigate human biases. The incorporation of AI into e-recruitment processes fosters a more efficient, cost-effective, and user-friendly candidate experience, ultimately providing organizations with a sustainable competitive advantage, [10]. The only Recommender System existing in the literature and capable of supporting online recruitment services is CASPER. This system operates in the following way: Once a user has been assigned, it classifies the job offers in relation to his needs and suggests job proposals based on his past behavior (in particular, it suggests proposals like those that he/she/it she has accepted in the past and avoids suggesting proposals similar to those that he/she has rejected previously), [11], [12].

The objective of this article is to advance the state of the art by updating a previously implemented initial work and improving its results; the aim is to propose a Recommender System created through intelligent agents and based on XML; the proposed system uses quite detailed user profiles to support the user in the personalized search for job offers. The system operates as follows: each user is associated with a User Agent (UA) that manages her profile. The system also features a Recruitment Agent (RA) which supports the various User Agents in selecting job offers relating to the corresponding users. This activity is carried out considering the user's past behaviors (suitably stored in her profile) and those interests not considered by him/her in the past, but which appear to be potentially interesting for him/her in the future. The selection of job offers is performed by means of an item-based recommendation algorithm. There are three elements that differentiate our approach from CASPER. First, our system is based on intelligent agents. The use of agents allows for improved autonomy, reactivity, and proactivity; furthermore, it allows workloads to be easily partitioned between the different entities involved, which significantly improves its scalability. Finally, intelligent agents simplify the conduction of existing legacy systems thanks to the definition of appropriate wrappers; consequently, our system can easily interact with existing online recruitment sites, and this helps to improve its accuracy, [13], [14], [15], [16], [17].

Secondly, the work proposal selection algorithm, which is the basis of our system, allows us to select the proposals considering not only the user's past behavior but also some proposals that he has not taken into consideration in the past, but which appear to be potentially interesting for him in the future, [18], [19], [20], [21], [22], [23].

Finally, our system is based on XML. XML is a standard for representing and exchanging data. XML combines the capability to represent data (typical of HTML) and the capability to manage data (typical of databases). An XML information source consists of a simple textual document; consequently, it is light and versatile, can be easily exchanged, and can be stored on various hardware and software platforms. Despite the simplicity of the language, the information representation rules present in XML are powerful enough to allow the conduction of sophisticated queries, [24], [25], [26], [27], [28], [29], [30], [31].

So, the objective of this paper is to demonstrate how this system can improve accuracy, scalability, and interaction with existing recruitment sites, providing advanced and personalized support to users in their job search. The research employs a combination of intelligent agent technology and XML-based data management to develop a robust and scalable recommender system. The proposed methods have been identified to make the most of the autonomy and reactivity of intelligent agents and the versatility and interoperability of the XML system. The selection of them is justified by the necessity for a highly scalable, efficient, and easy-to-integrate system that can communicate with online recruitment platforms already present online. The main scientific hypotheses are: the integration of intelligent agents in the recommender system will significantly improve the system's scalability and accuracy compared to existing systems like CASPER; using XML for data representation and exchange will enhance the system's ability to manage complex queries and improve interoperability with existing platforms; the inclusion of proposals not previously considered by the user but potentially interesting will increase user engagement and satisfaction with the job search process, [32], [33], [34].

2 Theory and Methodology

2.1 Intelligent Agent

Oriented Agents or Intelligent Agents represent one of the major innovations in the field of software engineering and Artificial Intelligence. In this article, Intelligent Agents and the agent-oriented programming paradigm are introduced by examining the distinct types of agents and the main purposes for which they were designed. An agent, software or robotic, is characterized by the ability to perceive its environment through sensors and act in the environment through actuators; it incorporates learning, reasoning, and planning capabilities and performs tasks for the benefit of a user or another agent. The presence of multiple agents, involved in a common mission, within the same physical or virtual environment, requires addressing coordination and collaboration issues. In particular, in highly dynamic environments, each agent can benefit from the use of models that describe the evolution, in space and time, of the environment and the state of the other agents (context). Maintaining these models can be further complicated by the incompleteness and inaccuracy of sensory data. An agent is uniquely defined by a PAGE, an acronym for Precept, Action, Goals, and Environment.

- Percept: the agent must be able to collect data from the environment around it, know where it is, and how to move within the environment itself
- Action: what an agent can perform.
- Goals: objectives that the agent intends to pursue
- Environment: environment from which it draws data and in which it acts.

We can define an agent as a set of software and hardware components that act to achieve objectives established by the user. In reality, the term agent is used to indicate a broad class of research and development projects; the same researchers, even in the field of Artificial Intelligence, have not yet defined general guidelines to which the design of a system of agents must comply.

2.2 Agent's Characteristics

An agent has crucial capabilities contributing to its effectiveness across various tasks. Communication stands as a fundamental attribute, allowing the agent to interact with other agents within the system and users alike. This proficiency facilitates information exchange, instruction reception, and the provision of guidance or assistance, as necessary. An agent is also distinguished by its capacity for autonomous action. Unlike passive advisors, it can independently execute a diverse array of operations, ranging from basic tasks to complex actions.

The versatility of an intelligent agent allows it to anticipate and satisfy users' needs and wants without the need for direct intervention. A fundamental aspect of these agents is their autonomy, which allows them to operate alone and independently, make decisions, and carry out tasks without the need for continuous supervision by the user. This autonomy translates into quick and efficient responses, allowing the agent to quickly adapt to changing circumstances and improve outcomes. Furthermore, an intelligent agent uses the accumulated experience to improve decision-making processes, thanks to the analysis of data and feedback from past interactions with which it can obtain valuable information on user preferences, behaviors, and patterns. This allows it to guarantee personalized advice, anticipate needs, and offer tailor-made solutions that meet user expectations. The main characteristics of an intelligent agent include adaptability and the ability to learn in a fast and personalized way. Despite the various definitions proposed by experts, there is a unanimous consensus on some essential properties to classify a software entity as an agent:

- Autonomy: This is an important property, rooted in the very concept of the agent since its birth. Both the user and the machine can initiate actions and monitor events. Software that provides this type of human-computer interaction is called autonomous. Autonomous agents use their knowledge of users' needs and interests to perform repetitive tasks. This quality is a difficult property to obtain, especially because it is strongly dependent on the characteristics of the content in which the program is operating. For human agents, the level of autonomy is clearly defined by laws, customs, or conventions. Indeed, some kind of agents vary their degree of autonomy based on a series of factors. Related to the concept of autonomy is the concept of proactivity.

- Adaptivity: adaptive agents must be able to learn through interaction with the external environment. This can include the physical world, human users, other agents, or the Internet. For this reason, adaptive agents are often called learning agents. The qualities necessary for an adaptive agent are:
- Reactivity: the agent perceives its environment and responds promptly to the changes that have occurred.
- Social ability: the agent interacts with other agents through distinct types of communication languages. The purpose of an adaptive agent is to adapt to the user, i.e., to learn the user's preferences. Adaptive behavior can be obtained in numerous ways, the simplest of which is to consider a set of users who perform the same operations and deduce their own behavior.
- Collaborative behavior: it is a typical characteristic of a set of agents, each of whom is assigned a specific task, but who collaborate in the pursuit of a common goal; they are often performed in parallel. Common problems in this context are establishing which agent must perform a certain task, integrating the information collected, and presenting it to the user. Agents must be able to communicate with other agents via appropriate communication languages, and being able to share a knowledge base is a crucial point in building cooperative agent systems. Since knowledge is shared, it must be represented in such a way that all agents can access it. The solution adopted is a representation conceptualized through ontologies.
- Mobility: refers to the ability of agents to migrate in a “self-directed” manner from one host to another in a network, to fulfill their as-signed tasks. This concerns finding information about each host, balancing traffic in the network, and so on. Mobility can be considered as an extension of the original concept of autonomy.

Combining all these definitions we can define intelligent agents of robots or software programs that are autonomous, adaptive, collaborative, and mobile.

2.3 An Ideal Internet Agent

To be considered optimal, an agent needs a set of specific capabilities. It must exhibit autonomy and adaptability, alongside the ability to interact effectively with users. This interaction necessitates a shared language with common vocabulary and syntactic structures. The primary challenge with existing search engines lies in their linguistic nature without a deep understanding of specific domains. This issue can be addressed by employing the concept of 'ontology' in Artificial Intelligence. Ontology formalizes knowledge by structuring it through class and subclass relationships. An ideal agent should demonstrate competence, enabling it to make informed decisions and actively provide services rather than merely offering advice. However, the imperative of user privacy often clashes with the conventional structure of the Internet, which typically requires access to personal data for specific services.

2.4 Programming of Intelligent Agent

An agent, from a legal point of view, is someone authorized to act in someone else's interest; an intelligent agent can learn the behavior of a user by watching the actions and transactions performed by him, in particular by observing his most repetitive behaviors. Intelligent agents are programs that function as a filter between the network user and the information arriving from them. In particular, they attempt to pick out the data that their respective user needs and, after having memorized the choices made by him over time, they will select, based on his profile, the information that best suits them. Each agent has a history, called a perception sequence, which keeps track of everything the agent has perceived through its sensors. For every possible sequence of perceptions, an ideal rational agent will perform the action that maximizes its performance. Once we have established that an agent's behavior relies only on its sequence of perceptions, we can characterize a particular agent by making a series of expected actions in response to each sequence of perceptions. In most cases, it is an exceedingly long table, not to say infinite, unless a limit is imposed a priori on the size. This list is called a mapping. We can uniquely describe an agent through mapping by trying out all possible sequences of perceptions and recording the actions in response. The problem is: how to build a program to implement perception-to-action mapping? This amounts to establishing how agents should use the data at their disposal to decide how to act.

There are four distinct categories within agent programs, each with its own approach to decision-making and action execution. Simple reflex agents operate according to a set of predefined rules. When they find themselves in a particular situation, they consult their repertoire of rules and, having found the appropriate rule, perform the corresponding action without further reflection. More complex than reflected agents are world-tracking agents. In addition to reacting to immediate stimuli, they maintain an internal state that allows them to adapt based on past experiences and the current state of the environment. By integrating past observations and feedback, this category of agents can make more refined decisions and change their actions accordingly.

Goal-based agents operate with specific goals in mind. Unlike reflex agents, which react only to immediate stimuli, these agents evaluate potential actions based on how much they contribute to achieving their goals. This approach allows them to make both reactive and proactive decisions, working towards achieving pre-defined goals. Utility-based agents use a utility function to evaluate the desirability of various states or sequences of states. They assign a numerical value (utility) to each state, reflecting how well it satisfies the agent's goals. By quantifying the desirability of different outcomes, they can navigate complex situations by resolving conflicts between competing goals and selecting actions that lead to the most promising outcomes. These four types of agent programs offer distinct strategies for decision-making and action execution. Each approach has its strengths and limitations, shaping the behavior of agents in diverse ways and enabling them to navigate diverse environments effectively.

2.5 Framework of Intelligent Agent

An Agent Program is a function that implements agent mapping based on perceived actions. An agent program can run on some architecture, hardware, or software; the architecture must provide a degree of isolation between the actual computer and the agent program so that high-level programming is possible. In general, the architecture makes the data perceived by the sensors available to the agent program, executes the agent program, and transmits the actions decided by it to the actuators. The main differences between programs and agents are that the program is: static, the actions are initiated by the user, non-interactive (the dialogue is written a priori), non-persistent (must be called to run again, once stopped), predictable (does what is told), follows instruction and stay in one place; while the agent is: dynamic, its actions can be initiated both by user or by agent system, can interact with users or other agents, is able to learn and to adapt, is persistent, is able to interpret what is means instead of what is said, can initiate action and is able to move through other servers.

2.6 Reasoning Techniques for Agents

It has been seen that the agents must be able to act autonomously and provide the user with the requested response. For this purpose, there are different reasoning techniques that are used.

Rule-based reasoning techniques: serve as the foundation for inference engines, allowing users to define rules or agent systems, often termed production rules in Artificial Intelligence. However, two significant challenges arise: firstly, users need to manually maintain data since these systems lack autonomous adaptation; secondly, the expansion or modification of the rule set may lead to conflicts, rendering the agent incapable of resolving them.

Knowledge-based reasoning techniques: These techniques, tailored to specific domains, underpin inference mechanisms, especially those utilizing rule-based approaches. Developers furnish agents with domain-specific task information, allowing them to deduce suitable responses based on their knowledge when confronted with particular situations. The main problem with these systems is that they require a large amount of work required. The best example in this application area is Cyc, a formalized representation of a vast amount of common human knowledge. This knowledge base can be used to build agents, each of which shares a common core of knowledge.

Simple statistical analysis: the simplest learning technique that an intelligent agent can use is statistical analysis. It can be used to determine the temporal correlation, if any, between events of interest.

Fuzzy agents: when an agent must work with partial and perhaps imprecise information, a useful tool is fuzzy logic; this is a form of logic used in various ex-pert systems in which variables can be classified based on truth or falsity values represented by values between 0(true) and 1(false).

Evolutionary Neural Networks: neural networks consist of a series of interconnected nodes, each with its own weight. The most popular type of neural networks are feed-forward networks; these consist of an input layer, an output layer, and a hidden layer; each of these layers consists of one or more processing units (neurons); each unit in a given layer connects all units in neighboring layers but not those in the same layer; each unit receives input from the units of the lower layer and sends output to the units of the upper layer. Weights are assigned by neural networks to connections between units, signifying the strength of the link. Agents that simply use neural networks, however, are limited to learning locally from certain documents or web pages. An agent's ability to explore is nevertheless restricted by the vast amount of information available online.

2.7 Environments

Agents operate in different environments, whose particular characteristics significantly influence their behavior and the decision-making processes that involve them. These environments can be identified and classified in distinct ways, highlighting the particularly complex dynamics that come into play in agent interactions with the surrounding environment. A fundamental aspect of environmental classification is the concept of accessibility, which refers to the presence of the agent's sensors that can identify, receive, and collect all aspects inherent to the choice of a certain action. In contrast, an inaccessible environment presents complications that influence the action of the agent's sensors, making it necessary to maintain the internal state for informed decision-making. Furthermore, it is important to underline how the quality of the action taken by the agent within each episode depends entirely on the characteristics of that particular episode, while subsequent episodes are completely independent of past actions. Non-deterministic environments, in contrast, introduce an element of randomness or uncertainty into the outcome of actions, making decision-making a more complex effort. The quality of the action within an episodic setting is determined solely by the unique characteristics of each episode, unprovided of any influence from previous episodes. In contrast, non-episodic environments have actions potentially having long-term consequences that are independent of individual episodes. Environments can then be classified as a function of dynamics: dynamic environments are characterized by their ability to submit alterations or transformations, introducing additional levels of complexity for agents as they navigate evolving circumstances. In contrast, static environments remain constant throughout the decision-making process, offering a more stable and predictable scenario. Semi-dynamic environments, on the other hand, remain overall unchanged but highlight variations in the agent's performance over time. Finally, environments can be classified as discrete or continuous, based on the nature of the data and actions available within them. Discrete environments present an established set of perceptible data and actions. Instead, continuous environments offer infinite possibilities. Therefore, the classification of environments along these various dimensions provides valuable insights into the challenges and opportunities that agents encounter in their respective domains. By understanding the shaded characteristics of different environments, researchers and practitioners can develop more effective agent architectures and algorithms tailored to navigate and thrive within specific environmental contexts.

2.8 Types of Use of Agents

Limiting ourselves to considering the case in which the environments are made up of computer networks, distinct types of use of the agents are observed.

•A first model was born as an evolution of the client-server architecture.

It can be “injecting” an agent into the network to perform a certain task, regardless of whether the connection with the client is interrupted. Injected agents navigate the network directly to service nodes, transforming client-server communication into local interactions. This approach minimizes network traffic and reduces the user's need for extensive service knowledge. Agents adopt a metaphor akin to market operations: visiting, using a service, and departing. Equipped with intelligence, these agents handle demanding tasks for users, configuring communication environments based on individual preferences. Users train agents to perform searches on their behalf by providing sample texts. The trained intelligent agent autonomously selects documents and web pages based on learned information. In the complex network landscape, using background agents to retrieve information aligns with a qualitative evolution. Unlike client-server relationships requiring numerous data transfers, agents, when injected, locally connect to the server, streamlining the retrieval of client-relevant data.

• Another agent model is used to automate repetitive tasks and learn user habits, acting as an intermediary to facilitate the execution of tasks within work processes. The greatest advantage of these agents is represented by the possibility of operating on behalf of the user even when he is not connected.

• An agent can function as a front-end for accessing a particular service. The provider must consider how potential users will use the new service when defining it. To interface with the service, the user needs to install a particular client on his node or use a generic browser. In the latter case, the server provider does not have the possibility of providing any added value to enrich and personalize the offer. To overcome this inconvenience, the service access front-end is defined by the service provider as an intelligent agent that migrates to the client node following a service request and which contains the logic necessary for interaction with the service itself. As we have just seen, the term "intelligent agent" is used with an overly broad spectrum of meanings: from adaptable user interfaces to communities of processes cooperating to achieve a common goal. For this reason, it becomes difficult to give a precise and exhaustive definition of the term.

As regards data collection and procedures, the data was collected through user interactions with the recommendation system. This includes accessing the system, searching for job offers, viewing recommended proposals, and interacting with graphical interfaces. Controlled experiments were conducted to evaluate the effectiveness of the recommendation system under different conditions or algorithm variations. Experimental conditions have included variables such as system usage frequency and user preferences.

As regards the conducted experiments, a representative sample of voluntary users with diverse professional backgrounds was selected. Subsequently, the database was populated with a significant number of job offers, selecting 1000 offers from www.jobpilot.it. The participants then took part in various sessions: 30-minute training sessions to explain the system and answer questions; 15-20 minutes profile collection sessions to allow participants to enter their data; 1-hour query execution sessions where participants could execute their queries, evaluate the offers, and provide feedback; 30-minute feedback and discussion sessions to gather qualitative feedback and discuss any issues encountered. Selection and interaction biases were managed to ensure that the sample of participants was representative and to standardize the training session for all participants so that everyone had the same level of understanding of the system.

3 General Architecture

The system consists of two main types of agents: a User Agent (UA), which manages the user's profile and its interaction with the system, and a Recruitment Agent (RA), which manages job offers. Finally, the system uses a Job Proposal Database (JPD) which stores data relating to job proposals. In the JPD, it can insertable inserted, remove or change the job offers with a precise Interface Agent. In addition, JPD can be automatically enriched by appropriate Wrapper Agents (like the one described which continuously `monitor' existing online recruitment sites, ex-tract new job proposals, and store them in JPD (if such proposals are not already present). JPD can be described by means of an XML document whose:

• JIDl is an identification code;

• JTopicSetl = {JTopicl1, JTopicl2, . . ., JTopiclm} is a set of `topics' describing JPl

• JCharacteristicSetl = {JCharacteristicl1, JCharacteristicl2, . . ., JCharacteristiclp} is a set of characteristics associated with JPl. Each feature is described by a pair feature, shown in Table 1.

Table 1. The possible characteristics associated with a job offer

Characteristic	*Description*	*Characteristic*	*Description*
Salary	The associate salary to job offers.	Mobility	Indicates whether the job offer requires the candidate to move to another location
Residence	The city of job offers.	Experiences	years of experience requested
Foreign languages	Foreign language required of candidates.	Academy	academic qualifications required of the candidate
Ability	Skills required of candidates	Type of job offer	Full-Time / Part Time job

In the recommendation system, the various agents play distinct roles, interacting to facilitate the personalized job recommendation process seamlessly. The User Agent (UA) serves as the intermediary between users and the system, managing user profiles and interactions. It collects user preferences and past behavior, communicating them to other system components. The Recruitment Agent (RA) is tasked with managing job offers, evaluating postings, and recommending relevant options based on user profiles. Using algorithms, it needs job ads from the Job Proposal Database (JPD) and gradually improves its recommendations. The Interface Agent, connecting the External subjects and the system, simplifies the process of addition, removal, and adaptation of jobs' offers inside the JPD; its incorporation with external origins and the consequent database update guarantees the integrity and the relevance of the data. The connections that occur between the agents are essential for the good working of the entire system and the process is the following: the users' request is sent to the RA, this one enters inside the JPD looking for relevant posts. At the same time, interaction occurs between the Interface Agent and the external authorities updating JPD. At the end of the process, through the User feedback collection, operated by UA, there are refines in profiles and better recommendations. In other words, these agents work together to provide a clear, customized job suggestion experience, increasing user engagement and happiness.

More in detail (Figure 1), at the beginning of the process, the operator (called "ui user") exposes its will to explore the eventual job's offers, sending an inquiry to their assigned User Agent, denoted as "Uai". Once received the user's request, the User Agent (Uai) starts to communicate with the system's Recommendation Agent (RA). In this phase, the User Agent, Uai transfers the user's question, correlated with the pertinent details, collected from the user's profile, allowing the foundation for the recommendation process. During the process, one of the most important functions of the system is the Recommendation Agent (RA), which is responsible for sorting through the job proposals stored in the Job Proposal Database (JPD). Using an advanced classification algorithm, RA determines if employment offers are appropriate by determining how well they match the user's profile. This judgment is based on a huge number of factors, such as the user's past actions and projected areas of interest, giving an accurate selection of job proposals related to the user's inclination and professional goals. These selected suggestions are then retransmitted back to the user agent (Uai) for submission to the user (ui), aiding informed decision-making. Once the proposals have been obtained, the Uai interacts with the user, showing him the personalized selection of work proposals to be evaluated and considered. Having a holistic view, the user can carefully evaluate each work proposal selecting the options that best fit his preferences and goals. Once the choice is complete, Uai merges the user's selection into their profile, keeping the system updated with the latest preferences and decisions. The system's recommendation capabilities are improved through this iterative profile refining process, which can lead to more customized and individualized suggestions. In the described process, the Naive Bayes classifier is the algorithm used for classification. When it is necessary to classify words or documents, such as user profiles and job advertisements like those that we have used in the case study, this kind of algorithm works especially well. Based on Bayes' theorem, the Naive Bayes classifier starts from the hypothesis that characteristics are conditionally independent of each other. In this case, the system was trained using a predictive variable, that is the characteristics of the user profiles and the job advertising, the aim was to assess the probability that a specific user is attracted to a specific job offer. Naïve Bayes classifier was applied to forecast the likelihood of a user being attracted to new job offers, according to offers' features and historical conduct. In our opinion, according to the scientific literature, the Naive Bayes classifier shows good computationally efficient properties, and it is easy to adapt to handle a large volume of data, making it appropriate for systems that work with large amounts of data and job offers are processed.

The use of XML in this system is an integral component, performing an important role in aiding the exchange of data within the different elements of the system. By leveraging XML as a standardized format for data encoding, the system adeptly transmits and interprets queries, user profiles, job proposals, and other pertinent information, thereby fostering efficient communication and enabling effective decision-making processes.

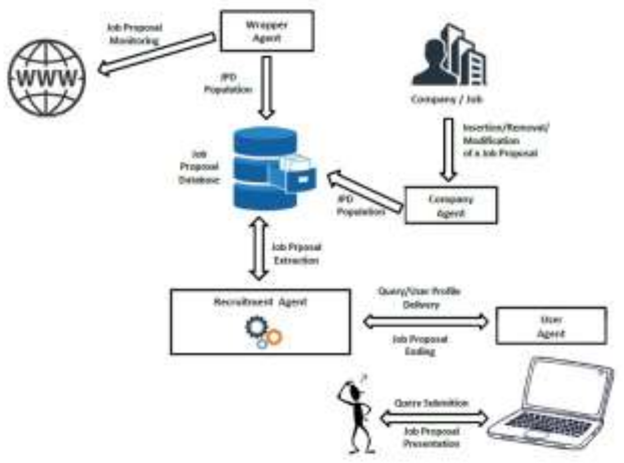


Fig. 1: General Architecture of the system

4 Results: Applicative Case Study

Using the described methodology and structure, this system was tested to verify its effectiveness in quickly matching users to the most suitable job opportunities by assessing the compatibility between the user's profile and job offers. The testing process consists of various steps:

1. User Profile Upload: when a user first logs in, the system requests them to upload their profiles, which should contain basic

information like job titles, experiences, jobs preferences.

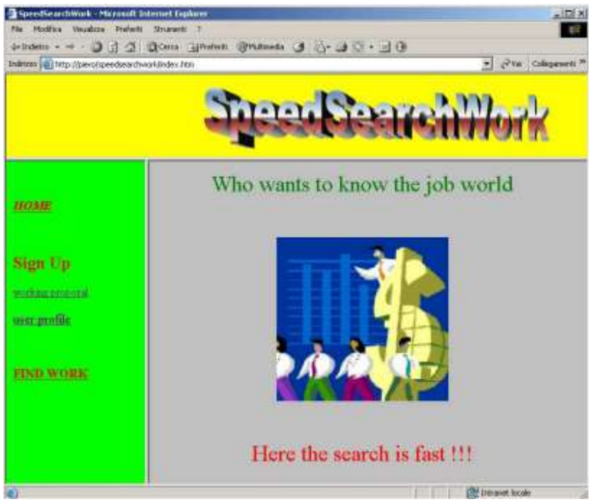
2. Job Offer Keywords: the system links each job offer to keywords or parameters to define their fields.
3. Combination of User Profile and Job Proposals: Using the Bayes' algorithm, the system flatters the suggesting process that occurs among user profile and job offers, examining the degree of correspondence among each job proposal, linking the keywords found in the previous step and the characteristics of the user profile.
4. Starting the search: after uploading the user profile and indexing the job with keywords, the research process is launched. This led to initiate queries to identify job opportunities that closely align with the user's characteristics and experiences.
5. Identification of suitable job offers: based on the alignment of the user profile with the job proposal keyword, the system finds and ranks the best job offers and the search goes further. These employment offers are prioritized and presented to the user.
6. Search storage: inside the system the research results are stored together with user's feedback. Thanks to this, the system is allowed to keep recorded research and results, simplifying future adjustments.
7. Search Refinement based on History: Leveraging the stored search data, the system continually refines and improves its search algorithms. By analyzing past search patterns, user preferences, and successful matches, the system adapts its recommendations to better align with user expectations and requirements.

The entire applicative case study is presented in the next subparagraphs.

4.1 Graphical user Interface

In the creation of this paper, we tried to define friendly graphical interfaces. Figure 2 shows the "entry" interface to the system. In particular, on the left side of the screen there is a menu divided into sections. The first section, "Sign up", allows the storage of a single job proposal or user in the system database. These interfaces adhere to fundamental design principles, such as user-centered design, consistency, simplicity, and feedback mechanisms: User-centered design is paramount, as it entails understanding user needs, behaviors, and goals to craft interfaces that align with their expectations; Consistency across design elements, including layout, typography, and color scheme, fosters familiarity and coherence, enhancing usability; An important characteristic of the system is the simplicity. In the proposed system, the implemented interface is based on an unassuming approach that shows the information without doubts and in a clear way, minimizing the cognitive burden. Users are driven to interactive objects and tools which guarantee that they can comprehend how to use the interface in the more useful way, using visual feedback mechanisms and clear pathways.

Usability of the proposed system also improves the client experience, emphasizing accessibility; in particular, users are helped during the research to find the wanted services, thanks to the system's user-friendly navigation paths and clear content organization. Several tools offered by the graphical interface, such as interactive objects, and intuitive input form, aid user engagement; Input forms are designed with clear labels and fields, inline validation, and autocomplete features to assist users in providing accurate information efficiently. Interactive elements, such as buttons and links, are visually distinct and responsive, encouraging user engagement.


SpeedSearchWork - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://piero/speedsearchwork/index.htm
Vai Collegamenti
SpeedSearchWork
Who wants to know the job world
HOME
Sign Up
working protocol
user profile
FIND WORK
Here the search is fast !!!
Intranet locale
Fig. 2: System input interface.

The next interface "Find work" is the most important part of the system and is described in the next paragraph. It displays the results obtained when a user executes a query. All interactions between the system and the user take place via Web pages. An example of interaction is illustrated in Figure 2. Through this page, it is possible to enter all the information that describes a user who accesses the system. Initially, the user is invited to fill in the form in the figure which is the interface of the XML document in Figure 3.

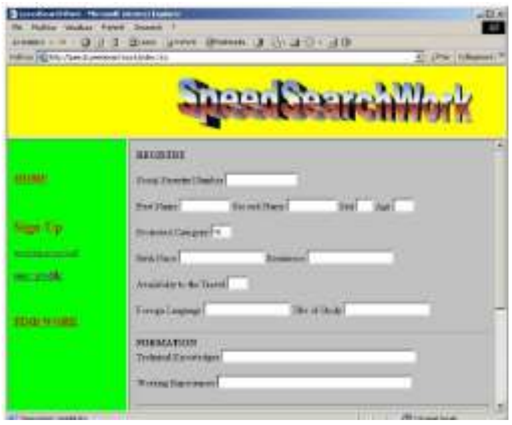


Fig. 3: Graphical interface for registering a user

```
[illegible]
```

Fig. 4: Empty XML knowledge base for a generic user

At the end of filling out the form, the system will provide a Username which will be used by the user to identify themselves whenever they wish to access the service. The information entered by the user is stored using appropriate XML documents. A generic organization that wants to insert one or more job proposals into the system has a graphical support interface at its disposal. This interface is shown in Figure 4. It graphically represents the XML document in Figure 5.

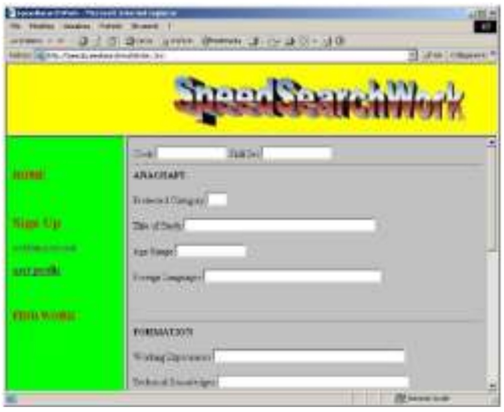


Fig. 5: Graphical interface for inserting new job proposals

```
[illegible]
```

Fig. 6: XML knowledge base for a generic job proposal.

The information associated with the single job proposal will be stored in an XML document, (Figure 6). To search for job offers of interest, a user has an appropriate graphical interface at his disposal. It is shown in Figure 7. The results following the execution of a query by the system are presented to the user via the graphical interface shown in Figure 8.

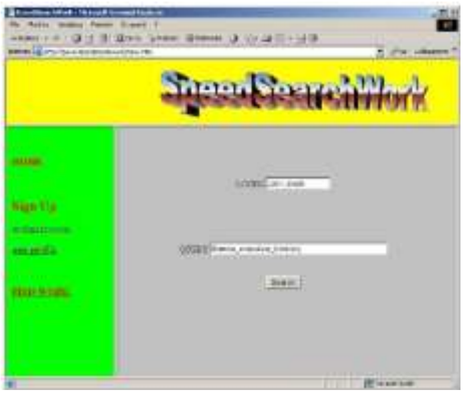
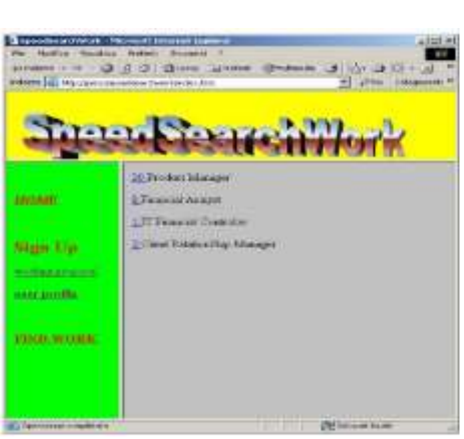


Fig. 7: Graphical interface for searching for job offers of interest to a user.



Product Manager
Financial Analyst
IT Financial Controller
Client Relationship Manager

Fig. 8: Results of a query.

Throughout this endeavor, the team focused on crafting intuitive and user-friendly graphical interfaces to facilitate seamless interaction between users and the system. The design objective was to provide effortless access to various functionalities such as user registration, job proposal insertion, and job search. For instance, the user registration interface allows users to input their information through a straightforward web form, structured in alignment with an XML document layout. Upon completion, users are assigned a username for future access, and their provided details are stored within XML documents. Similarly, organizations use a dedicated graphical interface to input new job proposals into the system, with each proposal's information structured within an XML document to ensure consistency and coherence. Leveraging XML as the data storage format offers versatility and interoperability, facilitating seamless data exchange across various systems and platforms. All interactions with the system occur through web pages, ensuring accessibility via standard web browsers and simplifying user engagement. The overarching aim of these interfaces is to establish a scalable and user-centric system supporting both users and organizations in the recruitment process, delivering a pleasant user experience and streamlining complex operations into clear and intuitive actions.

4.2 Experimental Analysis of the System Behavior

To evaluate the performance of our system, we performed several experiments. In them, the Recruitment Agent was "launched" on a Pentium IV with a 2.6 GHz processor and with 512 Mb of RAM, while the User Agents were activated on PCs equipped with a Pentium III processor with a 500 MHz frequency and with 128 MB of RAM. In our experimental campaign we used a set of users $U\,Set = \{u_1, u_2, \ldots, u_{50}\}$ made up of fifty volunteers. The available job offers were extracted from the website www.jobpilot.it; the number of available proposals is equal to one thousand. These proposals are associated with different application domains such as Information and Telecommunications Technologies.; Health Care; Legal and Administrative Sector; Marketing and so on. In our experiments we used two metrics widely used in the literature, namely Precision and Recall. In particular, each user $u_i \in U\,Set$ submitted a query Q_i and our system built the corresponding $JTempList_i$ set. Subsequently, we asked each user u_i to identify a set $UserList_i \in JPTempList_i$ of job proposals considered interesting. Finally, we activated our system to build the set $JPList_i$ The chosen metrics are: Precision and Recall. Precision and recall were chosen as evaluation metrics for the job recommendation system due to their ability to provide a comprehensive assessment of the system's performance in retrieving relevant job offers. In recommendation systems, there is often a trade-off between precision and recall. Focusing solely on one can adversely affect the other. Therefore, evaluating both metrics provides a balanced view of the system’s performance. Using both precision and recall ensures a comprehensive evaluation. Precision checks the relevancy of the recommendations, while recall ensures that no relevant options are overlooked. Their combined use helps in achieving a balanced and user-centric recommendation system. As an example, if our system retrieves 100 job offers for a user and 80 of these are relevant (based on user feedback), the precision is 80%. If there are 100 relevant job offers in the entire database and our system retrieves 80 of them, the recall is 80%. These metrics help us understand the balance between providing a sufficient number of relevant jobs offers and avoiding overwhelming the user with irrelevant options. The Precision Pre_i and Recall Rec_i associated with $u_i$were calculated as follows (1), (2):

$$Pre_i = \frac{|JPList_i \cap UserList_i|}{|JPList_i|} \quad (1)$$

$$Rec_i = \frac{|JPList_i \cap UserList_i|}{|UserList_i|} \quad (2)$$

The Average Precision $AvgPre$ and Average Recall $AvgRec$ associated with the set $U\,Set$ were calculated as follows (3), (4):

$$AvgPre = \frac{\sum_{i=1}^{|USet|} Pre_i}{|U\,Set|} \quad (3)$$

$$AvgRec = \frac{\sum_{i=1}^{|USet|} Rec_i}{|U\,Set|} \quad (4)$$

Before carrying out the experiments, we hypothesized that Precision and Recall should increase as the number of sessions carried out by users increases; in fact, our system was designed to adapt its behavior based on user feedback and their preferences. To verify this intuition, in the first series of experiments we calculated the values of $AvgPre$ and $AvgRec$ for different sessions of use of the system by the users. Figure 9 is the graphical representation of the experimental results.

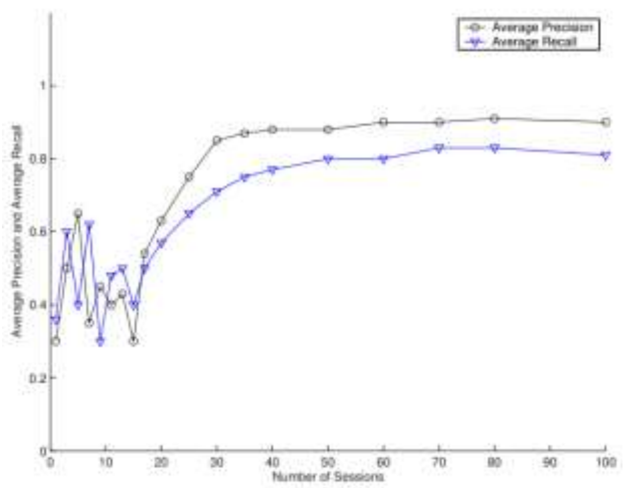


Fig. 9: Average Precision and Recall.

From this figure it is possible to observe that, initially, both $AvgPre$ and $AvgRec$ present an oscillatory behavior and take on rather low values in some sessions; for example, the minimum values of $AvgPre$ and $AvgRec$ are 0.3 and 0.36 respectively. This behavior is justified by the fact that, initially, both user profiles and their feedback are limited. However, after 30 sessions, both uupdateda previously implemented initial work and improving its results. and $AvgRec$ become almost constant and high; in particular, $AvgPre$ is always greater than 0.85 and avgrec is always greater than 0.75. These results confirm our original intuitions and indicate that the system performance is satisfactory. Finally, note that $AvgPre$ is generally greater than $AvgRec$; in fact, in this application context, precision has, in general, greater importance than Recall since the user could be "frustrated" by the presence of a high number of interesting proposals. As a subsequent experiment, we analyzed the impact of the initial value of the Audacity Degree R_0 on the system performance; in particular, we considered different values of R_0 and calculated the corresponding values of $AvgPre$ and $AvgRec$ after various sessions.

The corresponding results are shown in Table 2 and Table 3.

Table 2. Impact of R_0 on system performance.

Audacity	**Session 1**		**Session 5**		**Session 15**	
	AvgPre	AvgRec	AvgPre	AvgRec	AvgPre	AvgRec
0.05	0.5	0.3	0.42	0.65	0.36	0.46
0.15	0.34	0.44	0.46	0.52	0.33	0.45
0.30	0.38	0.41	0.53	0.5	0.36	0.34
0.40	0.3	0.36	0.65	0.4	0.34	0.4
0.60	0.5	0.42	0.55	0.61	0.32	0.43
0.75	0.43	0.32	0.51	0.5	0.32	0.41
0.90	0.52	0.45	0.42	0.46	0.35	0.35
1	0.43	0.52	0.40	0.44	0.33	0.33

Table 3. Impact of R_0 on system performance.

Audacity	***Session 30***		***Session 50***		***Session 100***	
	Avg Pre	Avg Rec	Avg Pre	Avg Rec	Avg Pre	Avg Rec
0.05	0.59	0.51	0.73	0.72	0.86	0.78
0.15	0.61	0.53	0.76	0.75	0.91	0.75
0.30	0.74	0.65	0.80	0.81	0.92	0.83
0.40	0.81	0.73	0.88	0.80	0.9	0.81
0.60	0.76	0.71	0.85	0.8	0.91	0.85
0.75	0.68	0.64	0.79	0.72	0.92	0.81
0.90	0.62	0.61	0.75	0.74	0.88	0.81
1	0.62	0.59	0.71	0.69	0.83	0.79

From the analysis of this table, we observe that if R_0 is too low or too high, or if it is less than 0.3 or greater than 0.6, our system requires numerous sessions to define the user's preferences and, consequently, to obtain acceptable $AvgPre$ and $AvgRec$ results. Conversely, when R_0 varies between 0.3 and 0.6, few sessions are needed (generally around 25) to obtain convincing results: for example, if R_0=0.4, after 30 sessions we have $AvgPre = 0.81$ and $AvgRec = 0.73$.

Results are also shown through plots in Figures 10a), 10b), 10c), 10d), 10e) and 10f).

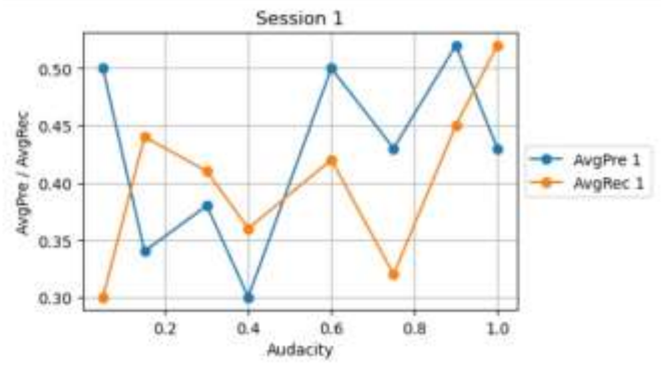


Fig. 10a: Session 1

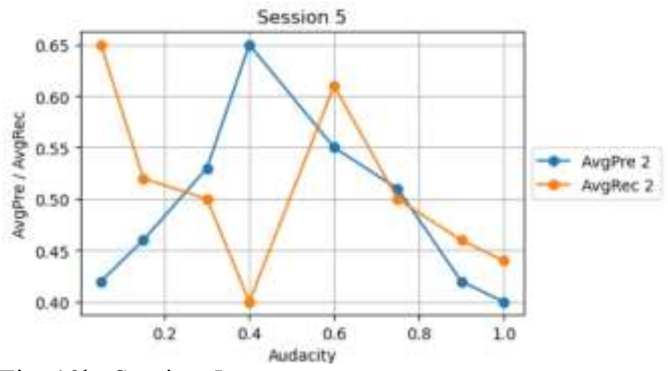


Fig. 10b: Session 5

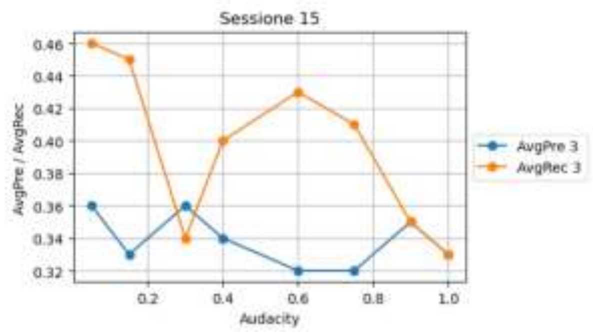


Fig. 10c: Session 15

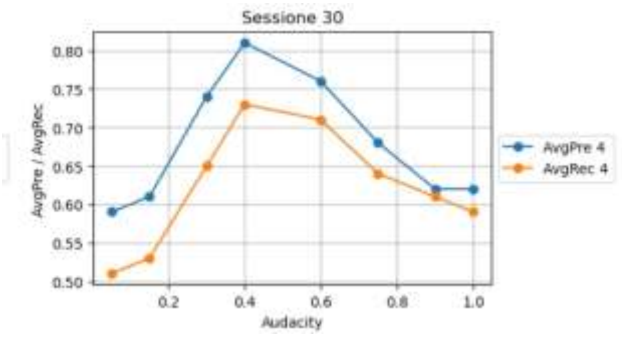


Fig. 10d: Session 30

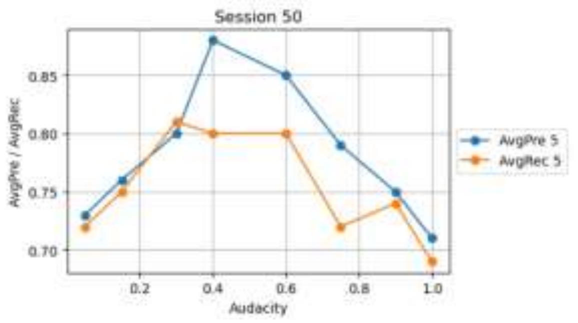


Fig. 10e: Session 50

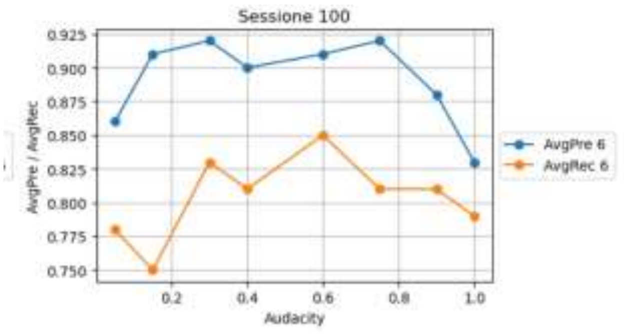


Fig. 10f: Session 100

Finally, to verify the performance of our system in different contexts, we applied our algorithm to data coming from different domains; some of them are generic, while others are extremely specialized. In this work we report the results obtained in two generic domains, namely Legal/Administrative and Marketing, and in two specialized domains, namely IT Consultant and Medical Consultant. For each of these domains, we measured $AvgPre$ and $AvgRec$ after several sessions. The results obtained are reported in Table 4 and Table 5.

Table 4. System performance in different application domains

Domain	***Session 1***		***Session 5***		***Session 15***	
	Avg Pre	Avg Rec	Avg Pre	Avg Rec	Avg Pre	Avg Rec
Legal-Administrative	0.5	0.3	0.71	0.65	0.83	0.75
Marketing	0.34	0.32	0.73	0.68	0.86	0.78
IT Consultant	0.44	0.5	0.42	0.39	0.6	0.45
Medical Consultant	0.4	0.5	0.28	0.4	0.54	0.41

Table 5. System performance in different application domains

Domain	***Session 30***		***Session 50***		***Session 100***	
	Avg Pre	Avg Rec	Avg Pre	Avg Rec	Avg Pre	Avg Rec
Legal-Administrative	0.87	0.81	0.78	0.75	0.75	0.73
Marketing	0.89	0.84	0.79	0.75	0.77	0.74
IT Consultant	0.7	0.58	0.86	0.75	0.94	0.89
Medical Consultant	0.66	0.6	0.83	0.73	0.92	0.9

From this table it is possible to observe that, in generic domains, our system obtains satisfactory results quickly; however, it has difficulty maintaining such performance after many sessions. Conversely, in specialized domains, our system requires many sessions to achieve satisfactory performance, however, after this initial phase, it can continuously guarantee outstanding performance. The explanation for this behavior is the following: during the first sessions, users of generic domains do not have a precise idea of their needs and consequently many suggestions proposed by the system appear of interest to them. Conversely, users of specific domains have a precise idea of their interests already during the initial phase and, consequently, are initially more difficult to satisfy. As the number of sessions increases, users of generic domains have a more precise idea of their needs, but the application domain they are interested in is too generic to satisfy their requests; however, from Table 4 and Table 5 it can be seen that, even in "hostile" conditions, the system achieves satisfactory performance. Conversely, after a high number of sessions, the profile of a user interested in a specific domain is quite rich and detailed and this allows our system to identify user needs with high precision and significantly reduce the search space for job offers. Finally, the future interests of users of specific domains can be predicted more easily than the future interests of users belonging to generic domains.

The results indicate that in generic domains, our system achieves high performance quickly but struggles to maintain this as the number of sessions increases. From this can be deduced that, although the first advice given by the system could be efficient, the system needs to be improved to manage long-term user operations without deteriorating its performance.

It is important to observe that, at the beginning and in specific domains, more sessions are needed by the system to achieve the best performance. But when adapted to the particular domains or sectors, for example, it constantly provides excellent results. This result shows the system's ability to know and adjust recommendations successfully in specialized sectors.

A paired t-test comparing our system's precision with that of CASPER revealed a statistically significant improvement ($p < 0.05$). The 95% confidence interval for the improvement in precision was [0.05, 0.15], indicating a robust enhancement in performance.

5 Conclusion

In this paper, an intelligent XML-based multi-agent system was proposed to support online worker recruitment services, updating a previously implemented initial work and improving its results. We have shown that our system consists of two main types of agents, namely: (i) a User Agent, which manages user profiles and interaction with the system, and (ii) a Recruitment Agent, which manages job offers. In addition, the system is enriched by two further types of agents, namely: a Wrapper Agent which finds new job proposals and inserts them into the system database, and a Company Agent, which allows companies to insert, remove, and modify job proposals. As regards the limitations, the recommendation system might be constrained by the availability and quality of historical data. If the data is incomplete or unrepresentative, it could negatively impact the accuracy of recommendations. The system's recommendations might not always align with users' true intentions or preferences, as the interpretation of user behavioral data can be subjective or occasionally inaccurate. Indeed, with the increase in users and job proposals inside the system, there may be some scalability issues regarding preserving systems's efficiency and handling massive data volumes. As a consequence, for the system's efficiency, it's mandatory that the data, especially that related to users' profiles and job postings, must be accurate, consistent, and complete. Addressing data quality issues may require rigorous data cleaning, preprocessing, and validation procedures. On the other hand, as regards the future directions, the proposed enhancements, including the integration of intelligent agents, the utilization of XML for data representation, and the implementation of an item-based recommendation algorithm, are technically feasible. Intelligent agents have been successfully applied in various domains, and there exist frameworks and tools to facilitate their development and integration. XML is a widely adopted standard for data representation and exchange, supported by libraries and APIs in most programming languages. Other potential future developments could be to introduce a multi-modal interaction between the agent and the user and to develop an adaptive schema evolution, able to dynamically adapt to changes in the data structure without requiring manual modifications to the XML schema, making the recommender system more flexible.

By leveraging intelligent agents and recommendation algorithms, the system can provide more personalized and relevant job recommendations tailored to individual user preferences. Improved user interfaces and intuitive interaction mechanisms can enhance the overall user experience, increasing user engagement and satisfaction. Automation of the job search process and targeted recommendations can save users time and effort, leading to improved efficiency and productivity. Furthermore, the scalability and adaptability of the system can be enhanced, allowing it to accommodate a growing user base and evolving user needs. As an evolution of the system, one could think of applying this methodology in any field of research, both in the documentary field, for example, the search for online documentation that reflects certain interests of the user, and in the planning field by relating the specific requests that a figure professional must have to be able to work internally on a project. Yet another evolution could include integrating the methodology with an e-learning system to build a system capable of suggesting to a user the knowledge that he/she should acquire to access new job offers.

The system could produce more accurate recommendations by understanding deeper into user preferences and job demand, through the analysis of textual and visual data.

In addition, improvements in speed and reliability could be achieved through the optimization of the system's performance, for example, using more performant algorithms, parallel processing techniques, and hardware acceleration.

In the future, additional tests utilizing novel AI algorithms might be conducted to improve the system's performance in terms of timing and outcome reliability. So, there is room for improvement in the suggested approach to boost the recommendation system's scalability, efficiency, customization, and user experience.

References:

[1] Moscarini, G., & Postel-Vinay, F. (2024). On the job search and business cycles. *Revue économique*, 75(1), 73-112. https://doi.org/10.3917/reco.751.0073 (Last accessed January, 2024).

[2] Hambarde, K. A., & Proenca, H. (2023). Information retrieval: recent advances and beyond, 11, 76581 - 76604. *IEEE Access*. DOI:10.1109/ACCESS.2023.3295776 (Last accessed February, 2024).

[3] Ramachandran, K. K., Phatak, S. S., Akram, S. V., Patidar, V., Raju, A. M., & Ponnusamy, R. (2023, January). Integration of machine learning algorithms for E-Learning System course recommendation based on Data Science. *In 2023 International Conference on Artificial Intelligence and Smart Communication (AISC)* (Greater Noida, India) (pp. 634-638). IEEE. DOI: 10.1109/AISC56616.2023.10085048 (Last accessed February, 2024).

[4] Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4), 35-43. (Last accessed February, 2024)

[5] Cesta, A., & D'Aloisi, D. (1996). Building interfaces as personal agents: a case study. *ACM SIGCHI Bulletin*, 28(3), 108-113. https://doi.org/10.1145/231132.231154 (Last accessed February, 2024)

[6] Deshpande M., Karypis G (2004). Item-based Top-N recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143-177. https://doi.org/10.1145/963770.963776 (Last accessed February 2024)

[7] Bradley, K., & Smyth, B. (2003). Personalized information ordering: a case study in online recruitment. *Knowledge-Based Systems*, 16(5-6), 269-275. https://doi.org/10.1016/S0950-7051(03)00028-5 (Last accessed March, 2024)

[8] Bankins, S., Ocampo, A. C., Marrone, M., Restubog, S. L. D., & Woo, S. E. (2024). A multilevel review of artificial intelligence in organizations: Implications for organizational behavior research and practice. *Journal of Organizational Behavior*, 45(2), 159-182. https://doi.org/10.1002/job.2735 (Last accessed March, 2024)

[9] González- Briones, A., Chamoso, P., Pavon, J., De La Prieta, F., & Corchado, J. M. (2024). Job offers recommender system based on virtual organizations. *Expert Systems*, 41(2), e13152. https://doi.org/10.1111/exsy.13152 (Last accessed March, 2024)

[10] Gusain, A., Singh, T., Pandey, S., Pachourui, V., Singh, R., & Kumar, A. (2023, March). E-Recruitment using Artificial Intelligence as Preventive Measures. *In 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS) (Erode, India)* (pp. 516-522). IEEE. DOI: 10.1109/ICSCDS56580.2023.10105102 (Last accessed February, 2024)

[11] Lieberman, H. (1995). Letizia: An agent that assists Web browsing. *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924 929, Montreal, Quebec, Canada. https://cdn.aaai.org/Symposia/Fall/1995/FS-95-03/FS95-03-016.pdf (Last accessed February, 2024)

[12] Liu, F., Yu, C., Meng. W. (2004). Personalized Web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):28-

40. DOI: 10.1109/TKDE.2004.1264820 (Last accessed February, 2024)

[13] Jennings, N.R., He M., Leung, H. (2003). On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):985-1003. DOI: 10.1109/TKDE.2003.1209014 (Last accessed February, 2024)

[14] Hsinchun, C., Yi-Ming, C., Ramsey, M., & Yang, C. C. (1998). An intelligent personal spider (agent) for dynamic Internet/Intranet searching. *Decision Support Systems*, 23(1), 41-58. https://doi.org/10.1016/S0167-9236(98)00035-9 (Last accessed February, 2024)

[15] Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3), 56-58. https://doi.org/10.1016/j.physrep.2012.02.006 (Last accessed February, 2024)

[16] Zaiane, O. R. (2002, December). Building a recommender agent for e-learning systems. *In International Conference on Computers in Education, 2002. Proceedings. (Auckland, New Zealand)* (pp. 55-59). IEEE. DOI: 10.1109/CIE.2002.1185862 (Last accessed February, 2024)

[17] Barrile, V., Cotroneo, F., Genovese, E., & Bilotta, G. (2023). Using Snn Neural Networks Trained with High Resolution Data 2 and Applied to Copernicus SENTINEL-2 Data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48, 27-31. https://doi.org/10.5194/isprs-archives-XLVIII-2-W3-2023-27-2023 (Last accessed June, 2023)

[18] Barrile, V., Cotroneo, F., Genovese, E., Barrile, E., & Bilotta, G. (2023). An AI Segmenter on Medical Imaging for Geomatics Applications Consisting of a Two-State Pipeline, Snns Network and Watershed Algorithm. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48, 21-26. https://doi.org/10.5194/isprs-archives-XLVIII-2-W3-2023-21-2023 (Last accessed June, 2023)

[19] Demir, M., & Günaydın, Y. (2023). A digital job application reference: how do social media posts affect the recruitment process?. Employee Relations: *The International Journal*, 45(2), 457-477. https://doi.org/10.1108/ER-05-2022-0232 (Last accessed March, 2024)

[20] Castilla, E. J., & Rho, H. J. (2023). The gendering of job postings in the online recruitment process. *Management Science*, 69(11), 6912-6939. https://doi.org/10.1287/mnsc.2023.4674 (Last accessed March, 2024)

[21] Younis, O., Jambi, K., Eassa, F., & Elrefaei, L. (2024). A Proposal for a Tokenized Intelligent System: A Prediction for an AI-Based Scheduling, Secured Using *Blockchain. Systems*, 12(3), 84. https://doi.org/10.3390/systems12030084 (Last accessed March, 2024)

[22] Woods, S. A., Ahmed, S., Nikolaou, I., Costa, A. C., & Anderson, N. R. (2020). Personnel selection in the digital age: A review of validity and applicant reactions, and future research challenges. *European Journal of work and organizational psychology*, 29(1), 64-77. https://doi.org/10.1080/1359432X.2019.1681401 (Last accessed March, 2024)

[23] Sanchez, W., Martinez, A., Hernandez, Y., Estrada, H., & Gonzalez-Mendoza, M. (2023). A predictive model for stress recognition in desk jobs. *Journal of Ambient Intelligence and Humanized Computing*, 14(1), 17-29. https://doi.org/10.1007/s12652-018-1149-9 (Last accessed March, 2024)

[24] Frissen, R., Adebayo, K. J., & Nanda, R. (2023). A machine learning approach to recognize bias and discrimination in job advertisements. *AI & SOCIETY*, 38(2), 1025-1038. https://doi.org/10.1007/s00146-022-01574-0 (Last accessed March, 2024)

[25] Naudé, M., Adebayo, K. J., & Nanda, R. (2023). A machine learning approach to detecting fraudulent job types. *AI & SOCIETY*, 38(2), 1013-1024. https://doi.org/10.1007/s00146-022-01469-0 (Last accessed May, 2024)

[26] Gnehm, A. S., & Clematide, S. (2020, November). Text zoning and classification for job advertisements in German, French and English. *In Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science (Canada)* (pp. 83-93). DOI:10.18653/v1/2020.nlpcss-1.10 (Last accessed April, 2024)

[27] Gupta, C., Singh, R. K., & Mohapatra, A. K. (2023). Secure XML parsing pattern for prevention of XML attacks. *In Information and Communication Technology for Competitive Strategies (ICTCS 2022) Intelligent Strategies for ICT* (pp. 759-770).

Singapore. Springer Nature Singapore. https://doi.org/10.1007/978-981-19-9304-6_68 (Last accessed February, 2024)

[28] Ganeriwala, P., Chambers, C., Sen, C., & Bhattacharyya, S. (2023, August). Functional Reasoning of System Architecture in the System Modeling Language (SysML) With XML Representation. *In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Boston, Massachusetts, USA)* (Vol. 87295, p. V002T02A044). American Society of Mechanical Engineers. https://doi.org/10.1115/DETC2023-117193 (Last accessed February, 2024)

[29] Zhu, L., Wang, J., & Bai, L. (2023). A general characterization of integrating and querying heterogeneous fuzzy spatiotemporal XML data. *Earth Science Informatics*, 16(4), 3303-3321. https://doi.org/10.1007/s12145-023-01091-8 (Last accessed February, 2024)

[30] Bai, L., Jia, Z., & Liu, J. (2017). Formal transformation of spatiotemporal data from object-oriented database to XML. J. *Digit. Inf. Manage.*, 15(1), 1. https://dline.info/fpaper/jdim/v15i1/jdimv15i1_1.pdf (Last accessed February, 2024)

[31] Pananjung, M. B., & ST, T. E. W. (2023, September). Development of SQL Interface for Spatial Data Processing in XML Databases. *In 2023 IEEE International Conference on Data and Software Engineering (ICoDSE) (Toba, Indonesia)* (pp. 79-84). IEEE. DOI: 10.1109/ICoDSE59534.2023.10291265 (Last accessed February, 2024)

[32] Pahuja, V., Dubey, R., & Sharma, I. (2023, December). Machine Learning-Based Detection and Mitigation of XML SQL Injection Attacks. *In 2023 Global Conference on Information Technologies and Communications (GCITC) (Bangalore, India)* (pp. 1-6). IEEE. DOI: 10.1109/GCITC60406.2023.10426458 (Last accessed March, 2024)

[33] Bikaki, A., Peters, M., Krozel, J., & Kakadiaris, I. A. (2024). Building an open-source collaborative platform for migration research: A metadata modeling approach using XML. *Knowledge-Based Systems*, 299, 111823. https://doi.org/10.1016/j.knosys.2024.111823 (Last accessed January, 2024)

[34] Bilotta, G., Genoves e, E., Citroni, R., Cotroneo, F., Meduri, G. M., & Barrile, V. (2023). Integration of an Innovative Atmospheric Forecasting Simulator and Remote Sensing Data into a Geographical Information System in the Frame of Agriculture 4.0 Concept. *AgriEngineering*, 5(3), 1280-1301. (Last accessed March, 2024)

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used Gemini in order to improve the readability and language for the introduction of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.