Ingénierie des Systèmes d'Information
Vol. 28, No. 4, August, 2023, pp. 1039-1053
Journal homepage: http://iieta.org/journals/isi

A Multi-Agent Systems Approach for Optimized Biomedical Literature Search

Ayman Mohammad Odeh Mansour[1,2*], Mohammad Ali Ahmad Obeidat[3], Jalal Mohammad Yousef Abdallah[3]

[1] Department of Computer and Communications Engineering, College of Engineering, Tafila Technical University, Tafila 6611, Jordan
[2] Faculty of Computer Studies (FCS), Arab Open University (AOU), Amman 11953, Jordan
[3] Department of Electrical Power and Mechatronics Engineering, College of Engineering, Tafila Technical University, Tafila 66110, Jordan

Corresponding Author Email: mansour@ttu.edu.jo

https://doi.org/10.18280/isi.280424

Received: 2 March 2023
Revised: 10 July 2023
Accepted: 26 July 2023
Available online: 31 August 2023

Keywords:
Multi-Agent Systems (MAS), literature search, recommender system

ABSTRACT

The potential of Multi-Agent Systems (MAS) in tackling the complexities of biomedical literature searches has been increasingly recognized. This research delves into the application of MAS for the amalgamation of varied information sources and expertise, striving for a higher degree of accuracy and comprehensiveness in search results. A distinct MAS framework, designed and implemented specifically for biomedical literature searching, is introduced. In this framework, decentralized agents are employed, each bearing responsibility for specific tasks such as data collection, pre-processing, information retrieval, and result evaluation. A collaborative and communicative environment among these agents is fostered to augment the overall performance of the system. To bolster the accuracy and comprehensiveness of the search outcomes, a variety of information sources and expertise are incorporated within the MAS. This amalgamation of expert knowledge and domain-specific information serves to enhance the relevance and accuracy of the retrieved results. Evaluation of MAS performance is carried out through multiple criteria and metrics, providing insightful feedback for continuous improvement of the system. The research illuminates the potential advantages of utilizing MAS in the realm of biomedical literature searches. The MAS framework demonstrates enhanced scalability, flexibility, and reliability when compared to traditional centralized approaches. Furthermore, the framework accommodates the integration of diverse expertise, allowing for the customization of the search process based on specific requirements. In conclusion, this study emphasizes the merits of MAS in advancing biomedical literature search by converging multiple sources of information and expertise. The results underscore the capability of MAS to navigate inherent challenges, thereby delivering precise and comprehensive search outcomes.

1. INTRODUCTION

1.1 Recommender systems

Recommender systems, a class of personalized software tools, are designed to suggest products or services tailored to users' individual preferences and behavior. These systems are utilized extensively across various industries. A prototypical example is Amazon's recommender system, which uses users' purchase history, browsing behavior, and product ratings to generate personalized recommendations. By considering patterns of other users with similar preferences, the system can offer recommendations appealing to a broad user base, thereby enhancing the pertinence and utility of its suggestions.

Netflix, a leading online streaming service, employs a recommender system deploying both content-based and collaborative filtering. Content-based filtering uses movie and TV show descriptions and genres, while collaborative filtering leverages ratings provided by users. This recommender system has significantly contributed to user retention and acquisition, demonstrating the power of personalized recommendations.

Numerous other industries also benefit from recommender systems. Spotify, for instance, uses such a system to suggest songs based on users' listening history and preferences. In the realm of travel, platforms like TripAdvisor use recommender systems to suggest destinations, hotels, and restaurants based on users' past travel experiences and ratings. E-commerce platforms, such as eBay, make product suggestions based on users' browsing and purchase history. YouTube's recommender system suggests videos based on viewing history, subscriptions, and liked videos. Despite the diversity of these applications, all these recommender systems share a common goal: offering personalized, relevant recommendations to users.

The development and challenges of multi-agent recommender systems for biomedical literature retrieval are crucial topics in the advancement of this field. Recommender systems have evolved from traditional single-agent approaches to sophisticated multi-agent systems, which can integrate diverse information sources and expertise. However, the effectiveness of such systems is often hindered by several complex challenges that require innovative solutions.

One such challenge stems from the complexity and heterogeneity of biomedical data, which includes a vast range

of structured and unstructured data, such as text, images, and clinical data. Advanced techniques for data preprocessing, integration, and representation are required to incorporate these diverse data types into a unified recommendation framework. Furthermore, obtaining reliable and comprehensive datasets for developing accurate and effective recommendation systems presents a significant challenge.

Efficient communication and coordination among the agents in the multi-agent system is another challenge. Agents must share information, collaborate, and synchronize their activities effectively to ensure coherent and reliable recommendations. Developing robust communication protocols and coordination mechanisms that can handle the diverse and dynamic nature of biomedical literature data is essential.

Scalability and performance are critical challenges in developing multi-agent recommendation systems for biomedical literature retrieval. As the volume of biomedical literature grows exponentially, systems must be capable of handling large-scale datasets and delivering timely recommendations. Efficient algorithms, optimization of computational resources, and leveraging parallel processing techniques are key considerations for achieving scalability and high-performance capabilities.

Security and privacy concerns represent another layer of complexity. Biomedical literature often contains sensitive patient information, necessitating robust security measures to protect data privacy. Compliance with relevant regulations and standards, such as the Health Insurance Portability and Accountability Act (HIPAA), while maintaining the integrity and confidentiality of data, presents a formidable challenge.

Beyond these technical considerations, user experience is a critical factor in the success of recommendation systems. It is crucial to develop intuitive user interfaces, personalized recommendations, and interactive features that meet the specific needs of biomedical researchers and healthcare professionals. Effective visualization techniques, user feedback mechanisms, and adaptive interfaces can greatly enhance the usability and acceptance of multi-agent recommendation systems.

Addressing these challenges calls for a multidisciplinary approach, incorporating expertise from fields such as artificial intelligence, machine learning, data integration, information retrieval, human-computer interaction, and biomedical informatics. By confronting these complexities, it is possible to overcome the challenges faced by multi-agent recommendation systems for biomedical literature retrieval, thereby facilitating more accurate, comprehensive, and efficient access to biomedical knowledge.

1.2 Multi-Agent Recommender Systems: A comprehensive overview

Multi-Agent Recommender Systems (MARS) constitute an innovative class among recommendation systems, leveraging the collaborative potential of multiple agents for the generation of personalized recommendations. These agents may be software-based, human-oriented, or a hybrid thereof, each tasked with curating recommendations for a distinct subset of users or items, or both [1, 2].

One of the salient advantages of MARS is their superior competency to manage substantial and intricate data sets, while maintaining recommendation accuracy. This is achieved by partitioning data into more manageable subsets, with each subset allocated to a specific agent. This not only accelerates data processing but also enhances efficiency. Furthermore, domain-specific specialization of each agent can lead to an enriched user experience through more personalized recommendations [3, 4].

Another strength of MARS lies in their capacity to amalgamate information from multiple sources for recommendation generation. Such a system could, for instance, utilize a user's purchase history, ratings from other users, and item content to curate a personalized recommendation list. This multifaceted approach fosters a comprehensive understanding of user preferences, thereby facilitating the creation of superior recommendations.

However, the design of MARS is not without challenges. Ensuring coordination among agents to produce consistent and complementary recommendations is crucial. Furthermore, the system must be equipped to manage potential conflicts between agents and validate the trustworthiness of the final recommendations.

Despite these challenges, MARS hold substantial potential to transform recommendation practices by offering personalized, precise recommendations. While the design intricacies of such a system are complex, the potential benefits warrant MARS as a promising avenue for further research and development [5-9].

1.3 Biomedical Literature Search Systems: An essential tool in biomedical research

Biomedical Literature Search Systems (BLSS) are specialized tools that are integral to the biomedical research landscape. These systems are meticulously designed to enable researchers to efficiently retrieve pertinent articles and papers from a plethora of scientific journals and databases, thereby supporting and streamlining their research endeavors.

A paramount example of BLSS is PubMed, a comprehensive database maintained by the National Library of Medicine. PubMed grants access to an excess of 29 million citations encompassing biomedical articles, peer-reviewed journals, online books, and conference papers. This system enables researchers to conduct topic-specific searches, or to identify articles that have cited a particular paper.

Another noteworthy BLSS is Embase, which provides access to an impressive portfolio of more than 30 million biomedical articles and conference papers. Covering a broad spectrum of biomedical subjects, Embase is especially advantageous for literature searches related to pharmacology and toxicology.

The role of BLSS in propelling the field of biomedical research forward is undeniable. By facilitating rapid and efficient identification of relevant articles, these systems contribute to the acceleration of research processes and the augmentation of research outcomes.

In conclusion, Biomedical Literature Search Systems are indispensable assets for biomedical researchers. By providing access to vast repositories of information, they assist researchers in identifying necessary data to support their work. The substantial contributions of these systems to the advancement of biomedical research are evident, and their critical role in the future of the field is assured [10-12].

In the study [13], the focus is on related search recommendations, a pivotal component of modern search engines. This paper delves into the integration of recommendation systems with search engines and underscores

the significance of incorporating knowledge graphs, user context, and feedback to enhance search results. The relevance of this research to biomedical literature search lies in its emphasis on optimizing search recommendations based on user behavior and context.

Paper [14] explores the realm of text mining and trend detection, particularly within the context of social media, specifically Twitter. It underscores the importance of indexing content and employing preprocessing techniques to effectively analyze user-generated text data. Pertinently, in the context of biomedical literature search, similar techniques can be applied to process and extract insights from user-generated content, such as reviews and comments related to research articles and medical topics.

Authors of [15] investigates the intriguing intersection of emotion recognition via facial expressions and its application in music recommendation. While this paper primarily addresses music recommendation, the concept of emotion recognition through facial expressions can be extended to biomedical literature search. Users' emotional responses to medical research can significantly impact their preferences and recommendations, making emotion recognition an intriguing avenue for optimizing search results in this domain.

1.4 The role of Artificial Intelligence in Biomedical Literature Search Systems

Artificial Intelligence (AI) is increasingly being integrated into **Biomedical Literature Search Systems**, enhancing their functionality and improving the speed and accuracy of search results.

One significant application of AI lies in the deployment of Natural Language Processing (NLP) techniques. NLP algorithms are proficient in extracting pertinent information from scientific articles and categorizing it based on its relevance to the researcher's query. Consequently, more accurate and relevant results can be returned, reducing the time researchers need to locate required information.

Machine learning, a subset of AI, is also being harnessed to optimize the functionality of Biomedical Literature Search Systems. Machine learning algorithms can be trained to identify patterns within data and make predictions regarding an article's relevance based on the available information. This ability further enhances the accuracy and speed of search results.

Additionally, AI is being utilized to improve the user interface of Biomedical Literature Search Systems. Notably, AI-powered chatbots can assist researchers in navigating the system and provide prompt answers to their queries, enhancing the overall user experience.

Machine learning is particularly prevalent in the field of Biomedical Literature Search Systems, employed in various ways such as:

(1) Document Classification: Machine learning is used to categorize articles based on their content and relevance to a specific query, thereby improving the accuracy of search results.

(2) Recommendation Systems: Machine learning can devise recommendation systems that suggest articles based on the researcher's search history, reading habits, and preferences, enhancing the user experience and reducing the time spent in locating necessary information.

(3) Information Extraction: Machine learning facilitates the extraction of relevant information from scientific articles, reducing the time researchers spend manually reading through articles.

(4) Query Expansion: Machine learning can expand a researcher's query to include related terms and concepts, improving the precision of search results.

(5) Sentiment Analysis: Machine learning can perform sentiment analysis on scientific articles to identify positive or negative sentiment and categorize articles accordingly.

These are just a few examples of the multitude of ways machine learning is contributing to the evolution of **Biomedical Literature Search Systems**. By improving the speed and accuracy of search results, machine learning is making it significantly easier for researchers to find the information they need. As the field of **Biomedical Literature Search Systems** continues to rapidly evolve, the role of machine learning is poised to become even more critical.

1.5 The application of Multi-Agent Systems in literature and Article Search Systems

Multi-Agent Systems (MAS) are distributed systems composed of multiple autonomous agents that interact to achieve a common objective. Within the realm of search systems, MAS have been effectively employed to enhance search results and user experience. For instance, agents can be designated to specific tasks such as web crawling, indexing, and ranking. Each agent, responsible for a specific task, communicates with other agents to collectively complete the overall search process. This division of tasks fosters improved efficiency and scalability of the search engine.

Furthermore, MAS have been utilized to personalize search results. Agents can monitor the user's search history and preferences, leveraging this information to provide customized results. For example, if a user frequently searches for specific data, such as sports or news, an agent can present results that correlate with the user's interests. Additionally, MAS have been used to detect and counteract spam and fraudulent activities in search engines. Agents can monitor search results and user feedback, enabling the search engine to detect and exclude false or misleading information.

In the field of literature and article search systems, MAS play a pivotal role in enhancing the speed and accuracy of search results and improving user experience. The following examples illustrate the application of MAS in this context:

(1) Query Distribution: MAS can distribute the search query across multiple databases, search engines, and other information sources, thereby improving the accuracy of search results.

(2) Information Fusion: MAS can consolidate information from multiple sources to produce a unified set of search results, enhancing the accuracy of search results.

(3) Personalization: MAS can personalize the search experience for each user based on their search history, reading habits, and preferences, thereby improving the user experience.

(4) Collaborative Filtering: MAS can implement collaborative filtering algorithms that suggest articles based on the reading habits of other users with similar interests, thus improving the accuracy of search results and reducing the time researchers spend locating necessary information.

These examples underscore the myriad ways in which Multi-Agent Systems contribute to the optimization of literature and article search systems. By improving the speed and accuracy of search results, MAS are making it significantly more straightforward for researchers to find the

information they need. As the field of literature and article search systems continues to evolve rapidly, the role of Multi-Agent Systems is set to become increasingly significant.

2. THE DEVELOPED MULTI-AGENTS SYSTEM

A multi-agent system for biomedical literature search is a complex system that involves various components and agents working together to provide relevant and accurate search results to the user. The system (Figure 1) includes several key components, including the user interface, agent manager, search agent, relevance feedback agent, data management agent, knowledge base, parser agent, presentation agent, user agent, validation agent, and feedback agent. The user interface is the front-end of the system, where users can submit their queries and view the search results. It communicates with the other components to receive user requests and display the results. The agent manager is responsible for coordinating the activities of the other agents. It assigns tasks, manages communication between agents, and ensures that the overall system functions smoothly. The search agent is responsible for conducting the actual search in the biomedical literature. It retrieves relevant articles and information from the databases, applies various search algorithms and filters, and returns the results to the user. The relevance feedback agent is responsible for refining the search results based on the user's feedback. It collects information about the user's preferences, such as the relevance of certain articles, and updates the search algorithms accordingly. The data management agent is responsible for managing the data used by the system. It updates the databases with new articles, indexes the information, and ensures that the information is organized and accessible. The knowledge base is a repository of information and knowledge about biomedical literature. It is used by the search agent to provide additional context and insights to the search results. The parser agent is responsible for parsing the results of the search, extracting relevant information, and transforming it into a format that can be easily processed by other agents. The presentation agent is responsible for presenting the information extracted by the parser agent to the user in a clear and user-friendly format. The user agent represents the user of the system, and is responsible for sending requests to other agents and receiving information from them. The validation agent checks the validity of the search results, ensuring that they are accurate and relevant. The feedback agent allows the user to provide feedback on the search results, such as indicating which articles are relevant, or requesting additional information about a specific article. This feedback can then be used to improve the search process, such as by adjusting the search algorithms or changing the way that the results are presented. Overall, the multi-agent system for biomedical literature search provides a comprehensive and flexible framework for searching, analyzing, and presenting biomedical literature. By combining various components and agents, the system can handle complex and uncertain information, providing relevant and accurate results to the user.

A Fuzzy Inference Unit can be added to the multi-agent system for biomedical literature search described above by incorporating fuzzy logic into the search algorithm used by the search agent. The Fuzzy Inference Unit would use fuzzy set theory and membership functions to determine the degree of similarity between the user's query and the articles in the database. The membership functions would be used to assign a fuzzy membership value to each article, based on its relevance to the query. This fuzzy membership value would then be used to rank the articles and determine which articles should be included in the search results.

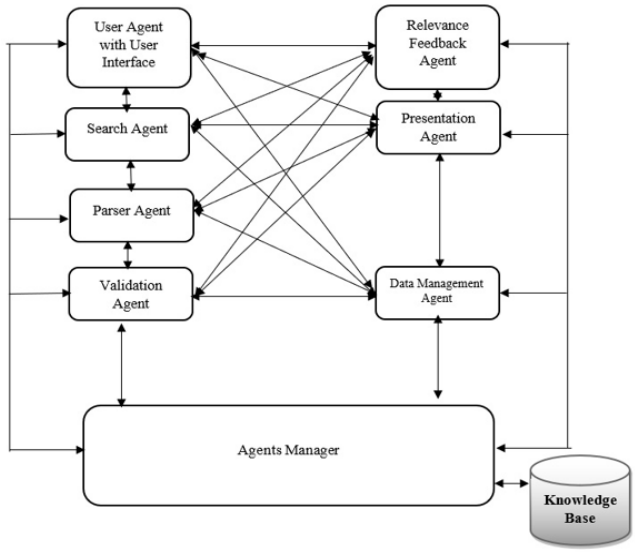

Figure 1. The develop multi-agent system

Additionally, the Fuzzy Inference Unit could also be used to refine the results based on the user's feedback. The relevance feedback agent would collect the user's preferences, such as the relevance of certain articles, and use these preferences to update the membership functions. The Fuzzy Inference Unit would then use these updated membership functions to refine the search results. This would allow the system to provide more accurate and relevant results, as it takes into account the user's preferences and provides a more nuanced approach to evaluating the similarity between the query and the articles.

Incorporating a Fuzzy Inference Unit into the multi-agent system for biomedical literature search would provide a more sophisticated and flexible approach to evaluating the similarity between the query and the articles in the database. This would result in more accurate and relevant results, and provide a better user experience.

The architecture of a Multi-Agent System (MAS) in a biomedical literature search can be divided into several components, each with a specific role and function. The main components are:

(1) Agent Manager: The agent manager is responsible for coordinating the activities of the other agents. It assigns tasks, manages communication between agents, and ensures that the overall system functions smoothly.

(2) Search Agent: The search agent is responsible for conducting the actual search in the biomedical literature. It retrieves relevant articles and information from the databases, applies various search algorithms and filters, and returns the results to the user.

(3) Data Management Agent: The data management agent is responsible for managing the data used by the system. It updates the databases with new articles, indexes the information, and ensures that the information is organized and accessible.

(4) Knowledge Base: The knowledge base is a repository of information and knowledge about biomedical literature. It is used by the search agent to provide additional context and insights to the search results.

(5) Parser Agent: This agent is responsible for parsing the results of the search, extracting relevant information, and transforming it into a format that can be easily processed by other agents. The parser agent may use text mining techniques to extract information from the search results.

(6) Presentation Agent: This agent is responsible for presenting the information extracted by the parser agent to the user in a clear and user-friendly format. The presentation agent may use a web-based interface, a graphical user interface, or another type of interface, depending on the requirements of the system.

(7) User Agent: This agent represents the user of the system, and is responsible for sending requests to other agents and receiving information from them. It includes user interface which is the front-end of the system. Through it the users can submit their queries and view the search results. It communicates with the other components to receive user requests and display the results.

(8) Validation Agent: This agent checks the validity of the search results, ensuring that they are accurate and relevant. The validation agent may also perform additional checks, such as checking for duplicates, or checking for missing or incorrect information.

(9) Feedback Agent: This agent allows the user to provide feedback on the search results, such as indicating which articles are relevant, or requesting additional information about a specific article. This feedback can then be used to improve the search process, such as by adjusting the search algorithms or changing the way that the results are presented.

These agents would interact with each other to accomplish the goal of the biomedical literature search system. For example, the user agent would send a search request to the search agent, which would send the request to the database or search engine, retrieve the results, and pass them on to the parser agent. The parser agent would then extract relevant information from the results, and pass it on to the presentation agent, which would present the information to the user.

Overall, the use of multiple agents in a biomedical literature search system provides a flexible and scalable architecture that can be adapted to different requirements and constraints, and that can handle large amounts of information in an efficient and reliable manner.

3. FUZZY SYSTEM IN BIOMEDICAL LITERATURE SEARCH IN THE DEVELOPED MULTI-AGENT SYSTEM

A fuzzy inference system (FIS) is a type of artificial intelligence system that uses fuzzy logic to perform decision-making tasks. The main idea behind fuzzy logic is to represent uncertainty and imprecision in a mathematical framework, allowing for more flexible and human-like reasoning compared to traditional Boolean logic. A FIS typically consists of several key components, including a fuzzification module, a rule base, an inference engine, and a defuzzification module. The fuzzification module converts the inputs into a fuzzy set representation, while the rule base consists of a set of if-then rules that specify the relationship between the inputs and outputs. The inference engine combines the inputs with the rules in the rule base to produce a fuzzy output, and the defuzzification module converts the fuzzy output back into a numerical or categorical representation. FISs are widely used in a variety of applications, including control systems, image processing, natural language processing, and biomedical literature search. The use of FISs in these applications is motivated by the ability of fuzzy logic to handle uncertainty and imprecision in a flexible and human-like manner, enabling the integration of expert knowledge and experience into the decision-making process. FISs provide a promising approach for addressing the challenges of decision-making in complex and uncertain environments, providing a flexible and human-like framework for reasoning and problem-solving.

Fuzzy systems are mathematical models that can be used to represent and process information with uncertainty. In biomedical literature search, fuzzy systems can be used to analyze and interpret the vast and complex data that is available in the biomedical field.

A fuzzy system in biomedical literature search can be used to process and interpret unstructured data, such as free-text medical reports and articles. The fuzzy system can use natural language processing techniques to extract relevant information from the text and assign a degree of confidence to each piece of information.

Fuzzy systems can also be used to make inferences based on uncertain data. For example, in biomedical literature search, a fuzzy system can be used to make predictions about the efficacy of a particular treatment based on the available data and research.

In addition, fuzzy systems can be used to develop recommendation systems in biomedical literature search. By analyzing user behavior and preferences, a fuzzy system can recommend articles and research that are likely to be of interest to the user.

The application process of fuzzy reasoning theory in the development stage of multi-agent systems is a crucial aspect that requires careful consideration. Fuzzy reasoning theory provides a framework for handling imprecise and uncertain information, which is particularly relevant in the context of multi-agent systems where diverse and uncertain data sources are integrated.

To effectively apply fuzzy reasoning theory, several steps are typically involved in the development stage of multi-agent systems. Firstly, the system needs to define fuzzy sets and linguistic variables to represent the imprecise or uncertain concepts relevant to the application domain. These linguistic variables can capture the nuances and vagueness inherent in the data.

Next, fuzzy rules are defined to establish the relationships between the linguistic variables. These rules encode expert knowledge or domain-specific information, enabling the system to reason and make intelligent decisions. The fuzzy rules are typically represented in the form of IF-THEN statements, where the antecedent (IF part) specifies the input conditions and the consequent (THEN part) defines the output or action to be taken.

The application process also involves the fuzzy inference mechanism, which applies the defined fuzzy rules to the input data and determines the appropriate outputs or actions based on fuzzy logic operations such as fuzzy matching, fuzzy aggregation, and defuzzification.

It is important to note that the application of fuzzy reasoning theory in multi-agent systems requires careful parameter tuning, membership function design, and rule optimization. These steps aim to ensure the effectiveness and accuracy of the fuzzy reasoning process in capturing and processing the uncertainty present in the system's data and decision-making.

Additionally, the application process may involve

evaluating and validating the performance of the fuzzy reasoning-based multi-agent system. This can be done through simulation studies, real-world experiments, or comparison with existing approaches to assess the system's effectiveness in achieving its objectives

3.1 Fuzzy Inference Units in biomedical literature search

Fuzzy Inference Units (Figure 2) are software agents that are designed to operate in a fuzzy environment, meaning an environment where information is uncertain and subject to interpretation. In biomedical literature search, Fuzzy Inference Units can be used to process and interpret complex data and make decisions based on that data.

Fuzzy Inference Units can be used to extract relevant information from biomedical literature, such as articles and research papers. By using natural language processing techniques, Fuzzy Inference Units can process unstructured data, such as free-text reports, and extract information that is relevant to the search query.

Fuzzy Inference Units can also be used to make inferences based on uncertain data. For example, a Fuzzy Inference Unit in biomedical literature search could use a fuzzy inference system to make predictions about the efficacy of a particular treatment based on the available data and research.

In addition, Fuzzy Inference Units can be used to develop recommendation systems in biomedical literature search. By analyzing user behavior and preferences, a Fuzzy Inference Unit can recommend articles and research that are likely to be of interest to the user.

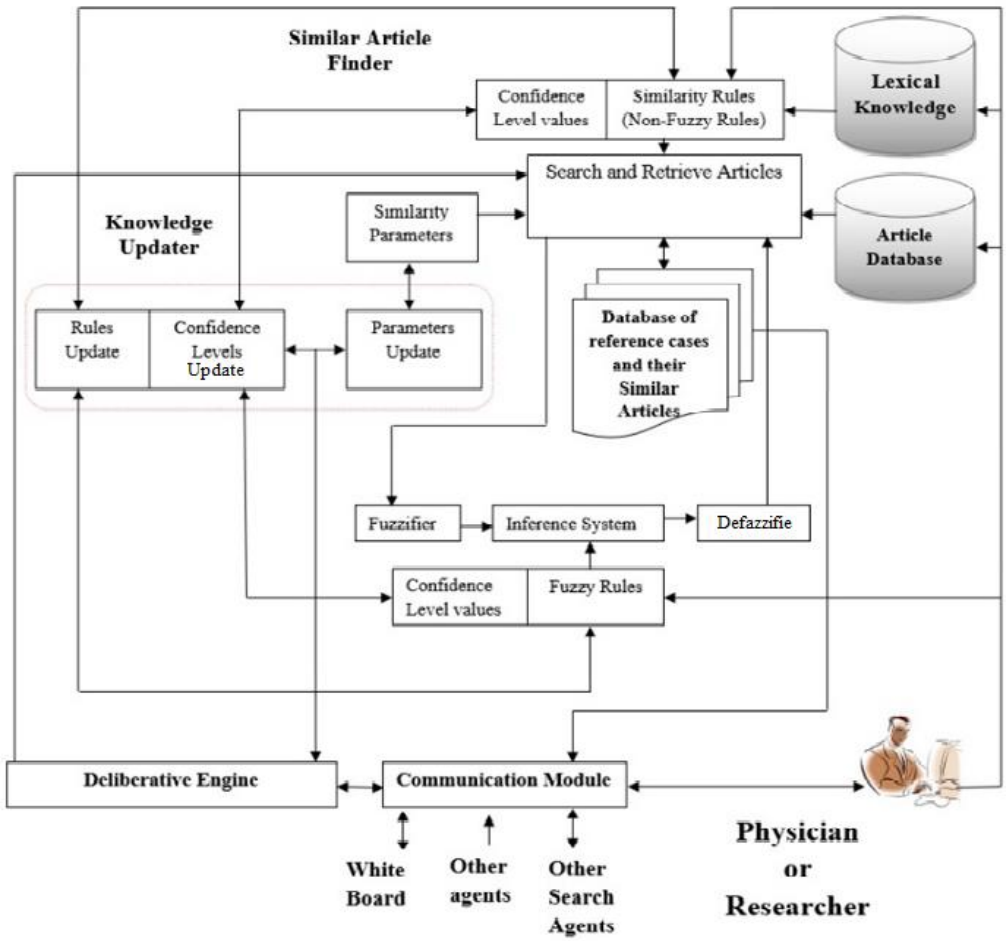

Figure 2. The architecture of the developed agents with fuzzy inference capability

3.2 Factors of fuzzy system in biomedical literature search

The factors affecting the performance of a fuzzy system in biomedical literature search can be grouped into two categories:

(1) Technical factors:

- Data quality and availability: The quality and quantity of data available for training the fuzzy system can greatly impact its performance.
- Feature selection: The choice of features to be used in the fuzzy system can greatly affect its performance.
- Algorithm design and optimization: The design and optimization of the fuzzy algorithm can greatly impact its performance.

(2) Domain-specific factors:

- Relevance and specificity of the query: The relevance and specificity of the query can greatly affect the performance of the fuzzy system in biomedical literature search.
- Bias in the data: Bias in the data used to train the fuzzy system can greatly affect its performance.
- Diversity of the literature: The diversity of the literature in the biomedical field can greatly affect the performance of the fuzzy system.

The multi-agent system utilizes a combination of fuzzy logic and text mining techniques in its filtering process to distinguish between articles based on their unique features, each of which may hold varying levels of importance. The

system takes into account factors that are not commonly found in traditional search engines, including medical cues and outcomes expected in the Results Section of an article, the use of Boolean AND and OR operators for cues and outcomes, weights assigned to keywords and features, and the presence of tables and figures. The articles are ranked according to their relevance to the user's search criteria, and the fuzzy filtering system comprises three subsystems: fuzzy keyword weights, fuzzy feature weights, and fuzzy relevancy ranking. The relevancy ranking indicates the degree of relevance each article holds for the user's search preferences. However, the weights assigned to keywords and features by different experts may differ due to disagreements, and therefore uncertainty exists within the obtained weights.

3.3 Membership function of articles similarity in biomedical literature search

In biomedical literature search, content similarity between articles is an important factor for determining their relevance to a given query. To quantify content similarity, a membership function is used to define the degree of similarity between articles.

A membership function maps the similarity score of two articles to a value between 0 and 1, where 0 represents no similarity and 1 represents complete similarity. The exact form of the membership function depends on the requirements of the specific fuzzy system being used. Some common forms of membership functions used in biomedical literature search include triangular, trapezoidal, and Gaussian functions.

The choice of membership function can greatly impact the performance of the fuzzy system in biomedical literature search. The membership function should be chosen based on the specific requirements of the task and the data being used. For example, a Gaussian membership function may be appropriate when the similarity scores have a Gaussian distribution, while a triangular membership function may be more appropriate when the distribution is less clear.

In general, the membership function should provide a smooth and continuous mapping of the similarity scores to the degree of membership, reflecting the uncertainty and imprecision that is inherent in content similarity in biomedical literature. The membership function should also be able to handle the variability and complexity of the biomedical literature data, capturing the relevant similarities and differences between articles.

When using fuzzy inference units to process parser proxy information, the membership function of period similarity is determined based on the specific requirements and characteristics of the system. The membership function for period similarity captures the degree of similarity or dissimilarity between periods.

The design of the membership function depends on factors such as the nature of the periods being compared, the desired granularity of similarity levels, and the available information for assessing similarity. Different forms of membership functions, such as triangular, trapezoidal, or Gaussian, can be utilized to represent the degree of similarity.

Regarding the multi-feature fusion process, it depends on the specific implementation and requirements of the system. Fusion of multiple features refers to combining different sources of information or features to make a decision or inference. In the context of processing parser proxy information, if the system incorporates multiple features related to period similarity, such as length, content overlap, or temporal distribution, then it may involve a multi-feature fusion process. The fusion process integrates the information from multiple features to derive a comprehensive assessment of period similarity.

It is important to note that the specific membership function for period similarity and the presence of a multi-feature fusion process depend on the design choices and requirements of the system. The membership function should be carefully defined to capture the desired notion of similarity, while the decision to employ a multi-feature fusion process depends on the availability and relevance of multiple features in the context of period similarity assessment.

3.4 Fuzzy system architecture in biomedical search

A fuzzy system architecture in biomedical search is a type of artificial intelligence system that uses fuzzy logic to perform biomedical literature search tasks. The architecture of a fuzzy system in biomedical search typically consists of several key components that work together to convert the inputs into a fuzzy set representation, process the data using a set of rules, and convert the output back into a numerical or categorical representation.

The first step in a fuzzy system is fuzzification, which involves converting the input data into a fuzzy set representation. This is typically done by mapping the numerical or categorical values to a set of membership functions that define the degree of membership of the inputs in different fuzzy sets.

The second component of the fuzzy system architecture is the rule base, which is a collection of if-then rules that specify the relationship between the input and output variables. The rules are used to process the fuzzy inputs and produce a fuzzy output.

The third component of the fuzzy system is the inference engine, which is responsible for combining the input data with the rules in the rule base to produce an output. The inference engine typically uses a combination of fuzzy logic operations, such as intersection and union, to process the data.

Finally, the fuzzy system architecture includes a defuzzification step, which involves converting the fuzzy output back into a numerical or categorical representation. This is typically done by mapping the fuzzy output to a single numerical value or a set of categorical values that represent the results of the biomedical literature search.

In the context of a multi-agent recommendation system, handling fuzzy system factors in biomedical literature involves incorporating fuzzy reasoning techniques to effectively address uncertainty and imprecision in the data. Fuzzy logic allows for the representation and manipulation of vague or ambiguous information, which is particularly relevant in biomedical literature where factors can exhibit varying degrees of fuzziness.

The sources of factor discrimination in a multi-agent recommendation system can include various aspects. Firstly, the linguistic variables used to represent factors may have different levels of granularity or membership functions. These linguistic variables capture the imprecision and uncertainty inherent in the factors under consideration. By defining appropriate membership functions, the system can discriminate different degrees or categories of factors, allowing for more nuanced recommendations.

Secondly, the fuzzy rules employed in the system play a

crucial role in discriminating factors. These rules establish the relationships between the linguistic variables, guiding the reasoning and decision-making process. Through these rules, the system can discriminate different combinations of factors and generate appropriate recommendations based on the available evidence.

Additionally, the sources of factor discrimination can also include the input data itself. Biomedical literature often contains complex and heterogeneous information, including textual data, numerical data, and categorical data. By considering these diverse sources of information, the system can discriminate factors based on their respective characteristics, such as the relevance, reliability, and significance of the information sources.

It is worth noting that the discrimination of factors in a multi-agent recommendation system involves not only handling the fuzziness of the factors but also considering their context and relevance to the specific biomedical domain. The system needs to carefully analyze and interpret the available data, taking into account expert knowledge, domain-specific considerations, and user preferences.

In summary, a multi-agent recommendation system handles fuzzy system factors in biomedical literature by incorporating fuzzy reasoning techniques to capture and process uncertainty. The sources of factor discrimination include linguistic variables, fuzzy rules, and the characteristics of the input data. By effectively discriminating factors, the system can generate relevant and accurate recommendations in the context of biomedical literature retrieval.

3.5 Block diagram of Multi-Agent System architecture in biomedical literature search

A block diagram of a multi-agent system typically consists of several key components, including:

(1) Agents: These are the individual components of the system that interact with each other to accomplish specific tasks. They are modeled as Java classes that implement the Agent interface in JADE, and they communicate with each other using JADE's messaging infrastructure.

(2) Message Broker: This component manages the flow of messages between agents. It is responsible for routing messages to their intended recipients, and ensuring that messages are delivered in a reliable and efficient manner.

(3) Agent Container: This component manages the individual agents and provides the runtime environment for executing the agents. It is responsible for starting and stopping agents, and for managing their behavior and interactions.

(4) Main Container: This is the core component of JADE, and is responsible for managing the overall operation of the multi-agent system. It is responsible for creating and managing Agent Containers, and for managing the interactions between agents.

(5) Data Store: This component stores information that is generated and processed by the agents. It may be a database, a file system, or another type of storage mechanism.

(6) User Interface: This component provides a user-friendly interface for interacting with the system. It may be a web-based interface, a graphical user interface, or another type of interface, depending on the requirements of the system.

These components interact with each other to perform the tasks of the multi-agent system. For example, an agent may send a message to another agent, which would be processed by the Message Broker and delivered to the recipient agent. The recipient agent would then process the message, generate a response, and send it back to the original agent. This process would continue until the desired goal of the system is achieved.

Overall, the block diagram of a multi-agent system provides a high-level view of the components and interactions that make

The architecture of an agent in a multi-agent system (MAS) typically consists of several key components, including:

(1) Agent Platform: This is the runtime environment for the agent, providing the necessary infrastructure for executing the agent's code and managing its interactions with other agents. JADE is one example of an agent platform that can be used to implement agents.

(2) Agent Behaviors: These are the individual components of the agent that implement its functionality. Each behavior can perform a specific task, such as receiving and processing messages, or making decisions based on data. Behaviors can be implemented as classes that inherit from the jade.core.behaviours. Behaviour class in JADE.

(3) Agent Beliefs: These are the beliefs or knowledge that the agent has about the world, such as information about other agents or information that has been learned through its interactions with the environment. Beliefs can be stored in data structures, such as arrays, lists, or maps, and can be updated based on new information.

(4) Agent Capabilities: These are the skills and abilities that the agent has, such as the ability to send and receive messages, or to make decisions based on its beliefs. Capabilities can be implemented as methods within the agent's behavior classes.

(5) Agent Actions: These are the actions that the agent can perform, such as sending a message to another agent, or updating its beliefs based on new information. Actions can be implemented as methods within the agent's behavior classes.

(6) Agent Communication: This component manages the communication between the agent and other agents. It provides the necessary infrastructure for sending and receiving messages, and for managing the flow of information between agents.

By dividing the agent into these components, the architecture of an agent can provide a clear and modular structure for implementing and managing the agent's functionality. This can make the agent more flexible and easier to understand, maintain, and extend over time.

4. MULTI-AGNENT SYSTEM COMMUNICATION TECHNIQUES AND SCENES

Multi-Agent System (MAS) communication is an integral part of the system's functioning, as it enables agents to interact and exchange information. There are several techniques and scenarios for communication in MAS, including direct communication, broker-based communication, and publish/subscribe communication. Direct communication involves agents exchanging messages directly with each other, while broker-based communication uses a central broker to manage the flow of messages. In publish/subscribe communication, agents subscribe to certain topics and receive updates when new information is published on those topics. The choice of communication technique depends on the specific requirements of the MAS, such as scalability, reliability, and security. Some common scenarios for MAS communication include negotiating tasks, exchanging information, and coordinating actions. In these scenarios,

agents must communicate with each other to achieve their goals and complete the overall system task.

Multi-Agent System (MAS) communication languages are used to facilitate communication between agents. These languages allow agents to exchange messages and coordinate their actions to achieve a common goal. Some common MAS communication languages include:

(1) FIPA-ACL (Foundation for Intelligent Physical Agents -Agent Communication Language): FIPA-ACL is a standardized language for agent communication, defined by the Foundation for Intelligent Physical Agents. It provides a standard syntax and semantics for representing messages, which makes it easy for agents to exchange information and interact with each other.

(2) KQML (Knowledge Query and Manipulation Language): KQML is a language for representing knowledge and exchanging messages in MAS. It allows agents to express complex information and request actions from other agents.

(3) Jason (Java Agent-Oriented Software Engineering): Jason is an open-source language for programming intelligent agents in Java. It provides a high-level language for specifying the behavior of agents, as well as a framework for managing communication between agents.

(4) TXL (Teaching Executive Language): TXL is a language for programming agents in a MAS. It provides a high-level syntax for specifying the behavior of agents, as well as a framework for managing communication and coordination between agents.

Agent communication can be presented in a number of ways, depending on the audience and the goals of the presentation. Some common methods include:

(1) Diagrams: Diagrams, such as flow charts or UML diagrams, can be used to visually represent the communication between agents. This can help to illustrate the flow of information and the interactions between agents.

(2) Examples: Presenting examples of actual communication between agents can help to demonstrate how the communication works in practice. This can be done by showing code snippets or sample messages exchanged between agents.

(3) Use Cases: Presenting the communication in the context of real-world scenarios, or use cases, can help to make the communication more relatable and easier to understand. This can be done by demonstrating how the communication between agents enables the agents to achieve specific goals.

(4) Simulation: Simulation can be used to demonstrate the communication between agents in a virtual environment. This can provide a visual representation of the communication and help to understand how the agents interact and exchange information.

(5) Demonstrations: Demonstrations can be used to show the communication between agents in real-time. This can provide a hands-on experience of the communication and help to understand how it works in practice.

Regardless of the method used, it is important to present the communication in a clear and concise manner, emphasizing the key elements and explaining any complex concepts. The presentation should be tailored to the audience, using terminology and examples that are appropriate for their level of understanding.

Diagrams are a useful tool for presenting agent communication, as they provide a visual representation of the flow of information and interactions between agents. Diagrams can help to illustrate the relationships between agents, the messages they exchange, and the protocols they follow. There are several types of diagrams that can be used to present agent communication, including flow charts, UML diagrams, and sequence diagrams. Flow charts are particularly useful for demonstrating the flow of information between agents, as they provide a high-level view of the interactions between agents. UML diagrams, such as sequence diagrams, can be used to show the detailed interactions between agents, including the messages they exchange and the timing of these interactions.

Diagrams can also be used to show the communication between agents in the context of a specific scenario or use case. This can help to demonstrate how the communication enables the agents to achieve a specific goal or solve a problem. Additionally, diagrams can be used to show the communication between agents in different phases of a task, or in response to different events. Overall, diagrams are a powerful tool for presenting agent communication, as they provide a visual representation of the interactions between agents. They can help to understand the flow of information, the relationships between agents, and how the communication enables the agents to achieve their goals.

A state diagram in multi-agent systems is a graphical representation of the different states that an individual agent can be in, and the transitions between those states. It is a type of finite state machine that models the behavior of an agent over time. In a state diagram, each state is represented by a circle and the transitions between states are represented by arrows. The transitions are labelled with the conditions or events that trigger them, and the diagram can also include additional information such as the actions that are performed when entering or leaving a state. State diagrams can be useful for modeling and analyzing the behavior of multi-agent systems, as they provide a clear and concise representation of the different ways that the agents can interact and respond to events in the environment. By using state diagrams as shown in Figure 3, it is possible to identify potential problems or conflicts between agents, as well as to design control algorithms that coordinate the behavior of the agents. Overall, state diagrams are an important tool for understanding and designing multi-agent systems, as they allow for a visual representation of the behavior of each agent and the interactions between them.

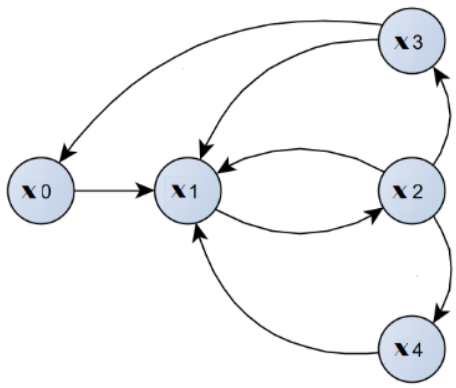

Figure 3. The state diagram of a scene in multi-agent environment

The automaton in Figure 3 has a set of states, including:

- X0: the initial state
- X1: waiting for a response to the request

- X2: checking the data
- X3: indicating lack of communication
- X4: processing the data.

Petri nets are a type of mathematical modeling tool used to describe and analyze complex systems, including multi-agent systems. In a multi-agent system, there are typically multiple autonomous agents that interact with one another and with their environment to achieve specific goals. Petri nets can be used to model the interactions between agents, as well as the environment they operate in, to better understand the behavior of the system as a whole.

Petri nets are composed of two main elements: places and transitions. Places represent states of the system, while transitions represent events or actions that can occur within the system. Tokens are used to represent the current state of the system and are placed within the different places. When a transition is enabled, meaning that all the places that lead to the transition have tokens, the transition can fire, moving tokens from the input places to the output places.

In a multi-agent system, each agent can be represented as a separate Petri net. The places and transitions within the agent's net represent the agent's state and possible actions. Interactions between agents can be modeled by connecting the input and output places of different nets. For example, as in Figure 4, if Agent A needs to receive a message from Agent B before it can take a certain action, the input place of Agent A's transition representing that action can be connected to the output place of Agent B's transition representing sending the message.

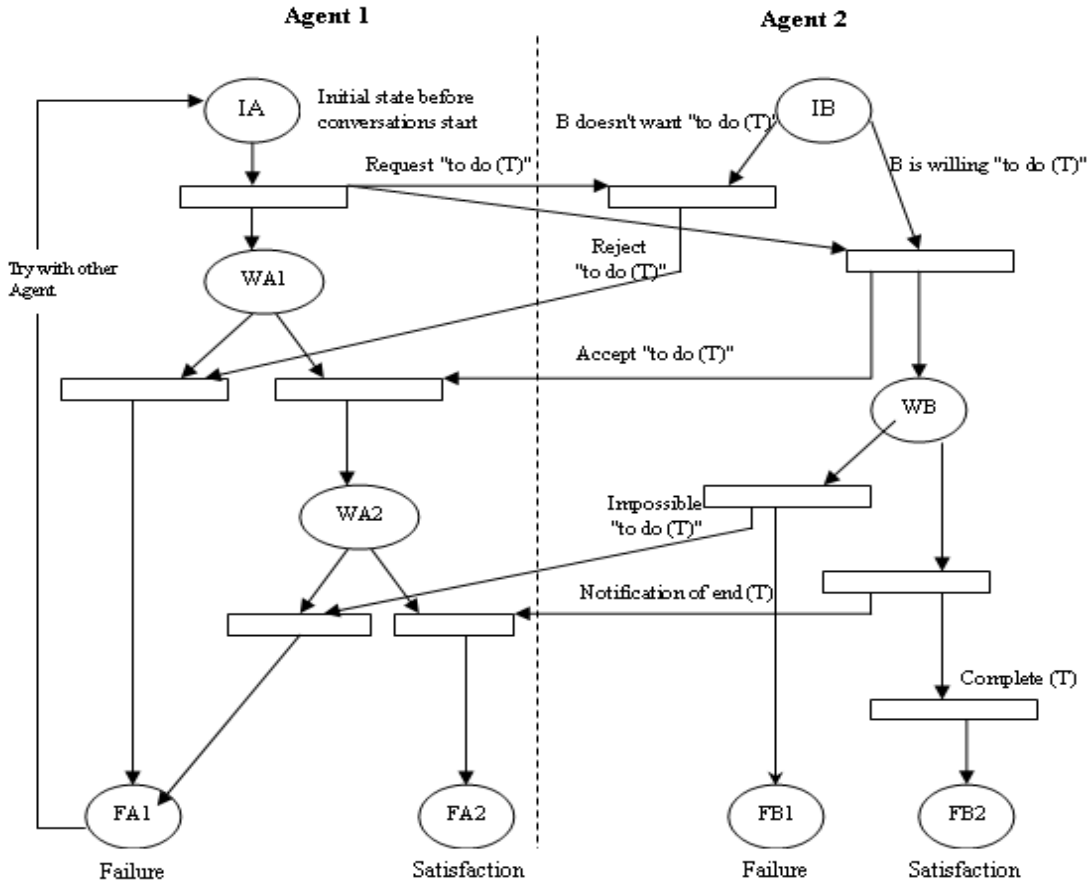

Figure 4. Conversational model of between two Agents using the Petri nets

The transitions correspond either to synchronization due to the receipt of messages or to conditions of actions. Places IA and IB describe the initial states where the PAAs find themselves before the beginning of the conversation. Places FA1, FA2, FB1 and FB2 represent the end of conversation states. Starting from state IA, Agent 1 sends a request “to do (T)” to Agent 2 and moves in to state WA1, which represents waiting for a response. If AGENT2 can’t do the task, it sends a refusal to Agent1, which then goes into state FA1, which indicates that Agent 1 must look elsewhere to have its task carried out. If Agent 2 can do the task, it sends an acceptance message to Agent 1, which places Agent1 into wait for a response state WA2. During this time, Agent 2 is in state WB, while trying to carry out the task. Once finishing the task, it sends a notification of end of accomplishment to Agent1, which places Agent1 in state FA2 and places AGENT2 in state FB2. If not, Agent 2 indicates that it cannot do the task, which places Agent1 in state FA1 and places Agent 2 in state FB1.

Petri nets can also be used to model the environment in which the agents operate. The environment can be represented as a separate Petri net, with places representing different states of the environment and transitions representing events or actions that can occur within the environment. For example, the environment may include a place representing the location of a resource that agents can access, and a transition representing the acquisition of that resource.

Petri nets can help to identify potential issues in a multi-agent system, such as deadlocks or live locks. Deadlocks occur when none of the agents can take any further action, while live locks occur when the agents continue to take actions but are unable to achieve their goals. By modeling the system as a Petri net, it is possible to identify the conditions that lead to these issues and take steps to address them.

5. MULTI-AGENT SYSTEM IMPLEMENTATION USING JADE FOR BIOMEDICAL LITERATURE SEARCH

Managing the complexity of multi-agent systems requires employing various strategies and techniques. One approach is to decompose the system into modular components or agents, dividing the overall complexity into more manageable units. Effective coordination and communication mechanisms are essential for seamless interactions among agents, ensuring information exchange and collaboration. Organizational structures and defined agent roles help distribute tasks and responsibilities, reducing complexity. Proper information and knowledge management, through techniques such as knowledge representation and sharing, streamline the handling of complex data. Granting agents autonomy and adaptability enables them to handle local complexities independently, simplifying system management. Simulation and modeling techniques allow for analyzing and understanding system behavior in controlled environments. Monitoring and analysis tools provide insights into system dynamics and performance, aiding in complexity management. By applying these strategies, developers and researchers can effectively manage the complexity of multi-agent systems, achieving efficient and robust behavior.

JADE (Java Agent Development Framework) is a software platform for implementing multi-agent systems (MAS) in Java. It provides a set of tools and libraries for developing, deploying, and running agents, as well as for managing the interactions between agents. In JADE, agents are implemented as Java classes that extend the jade.core.Agent class.

The Agent class provides a number of functionalities to the agents, including message handling, sending messages, and agent mobility. Additionally, JADE also provides a number of agent management functionalities, such as starting and stopping agents, creating and removing agents, and monitoring agent behavior.

JADE agents can interact with each other by exchanging messages, which are instances of the jade.lang.acl. ACLMessage class. These messages can be of different types, such as requests, proposals, or informations, and can also be sent with different communicative intents, such as informing, asking, or commanding.

Another important aspect of JADE is the concept of agent roles. Agents can assume different roles in different contexts, and can also change their roles dynamically. This makes it possible to model complex agent interactions and cooperation, as well as to implement flexible and adaptable systems.

The Agent interface is the core component of JADE, and defines the basic methods and behaviors of an agent.

JADE provides several built-in classes for creating and implementing agents, including:

(1) Base Agent: This is the basic class for creating an agent in JADE. It provides the default implementation of the Agent interface, including methods for sending and receiving messages, and managing the agent's behavior.

(2) Simple Agent: This is a subclass of Base Agent that provides a simplified API for creating and managing agents. It is particularly useful for simple, single-function agents that do not require advanced features or behaviors.

(3) Cyclic Behaviour: This class is used for defining the behavior of an agent. It provides a simple way to implement agents that perform periodic or cyclic tasks.

(4) One Shot Behaviour: This class is used for defining the behavior of an agent that performs a single action or task. It is particularly useful for agents that are created for a specific purpose, and then terminate after completing their task.

(5) Sequential Behaviour: This class is used for defining the behavior of an agent that performs a series of actions in a specific order. It is particularly useful for agents that need to perform a complex set of tasks.

(6) Parallel Behaviour: This class is used for defining the behavior of an agent that performs a set of tasks in parallel. It is particularly useful for agents that need to perform multiple tasks simultaneously.

In addition to these built-in classes, JADE also allows developers to create their own custom agents by subclassing Base Agent or one of its subclasses. This allows developers to create agents with custom behaviors and functionality that are tailored to their specific needs.

Overall, JADE provides a rich set of classes and tools for creating and implementing agents, allowing developers to create agents with a wide range of behaviors and functionality.

To implement an agent in JADE, you typically start by defining a class that extends the Agent class, and then implement the necessary functionalities, such as message handling and sending, and agent mobility. You can also use a number of provided behaviors to simplify common tasks, such as message handling and periodic tasks.

Implementing a multi-agent system using JADE typically involves the following steps:

(1) Designing the agents: In JADE, each agent is modeled as a Java class that implements the Agent interface. The agents can be designed to perform a specific task or role within the MAS, and can interact with other agents by sending and receiving messages.

(2) Defining the communication protocol: JADE provides a messaging infrastructure that allows agents to exchange messages and data with each other. The communication protocol defines the structure and format of the messages that are exchanged between agents.

(3) Setting up the environment: JADE provides a runtime environment for deploying and executing agents. This environment includes the Main Container, which is the core component of JADE, and the Agent Container, which is the component that manages the individual agents.

(4) Deploying the agents: Once the agents have been designed and the communication protocol has been defined, the agents can be deployed to the JADE environment. This involves creating instances of the agents and registering them with the Main Container.

(5) Interacting between agents: Once the agents are deployed, they can start interacting with each other by sending and receiving messages. The interactions between agents can be managed by JADE's messaging infrastructure, which handles the delivery and processing of messages between agents.

JADE provides a number of additional features and tools for developing multi-agent systems, including a graphical interface for managing agents and messages, a library of pre-built agents and behaviors, and support for distributed and mobile agents.

Overall, JADE is a powerful tool for implementing multi-agent systems, as it provides a comprehensive set of features and libraries for developing, deploying, and executing agents, and for managing the interactions between agents.

6. EXPERIMENT AND SIMULATION RESULTS

There are several factors that can impact the design of a multi-agent system for biomedical articles search. Firstly, the complexity and heterogeneity of the data sources and the information retrieval methods used by the agents can affect the design. Secondly, the level of collaboration and communication required between the agents can also influence the design, as well as the types of decision-making algorithms used to coordinate their efforts. Thirdly, the scalability of the system, its performance and its ability to handle large volumes of data, can affect the design, as well as the security and privacy requirements for the system. Additionally, the usability and user experience of the system can also play a role in the design, including the ability for users to easily access, interpret, and use the information retrieved by the agents. These are some of the key factors that need to be considered when designing a multi-agent system for biomedical articles search.

Testing a multi-agent system (MAS) for biomedical article search can be accomplished through various methods, including functional testing, performance testing, user acceptance testing, integration testing, and regression testing. The specific testing methods will depend on the requirements and goals of the system.

The settings for agent service description design, communication information, and termination of operation in a multi-agent system are specified through various mechanisms and protocols. Here is an overview of each aspect:

Agent Service Description Design: The agent service description refers to the specification of the services provided by each agent in the system. It involves defining the functionalities, capabilities, and parameters of the services offered by the agents. This can include details such as the input/output data formats, supported operations, required resources, and any constraints or limitations. The service description design can be standardized using common ontology or description languages to ensure interoperability and understanding among agents.

Communication Information: Communication among agents in a multi-agent system is crucial for coordination and collaboration. The communication information includes specifying the communication protocols, message formats, and message exchange patterns used for interaction. This can involve defining the message structure, message headers, and payload data. The communication information also encompasses the addressing scheme, identifying the agents involved in the communication, and establishing the necessary connections or channels for data exchange.

Termination of Operation: The termination of operation in a multi-agent system refers to the process of ending an agent's participation or shutting down the system. The specification of termination can include conditions or triggers that determine when an agent should terminate its operation. These conditions can be based on predefined criteria, such as completing a specific task, reaching a certain state, or receiving a termination signal from another agent or the system. The termination process may involve releasing resources, notifying other agents about the termination, and ensuring a graceful shutdown to maintain system integrity.

Functional testing verifies that each component of the system performs its intended function correctly. This can be done by creating test cases that simulate different user scenarios and comparing the system's response to expected results. Performance testing evaluates the system's speed, scalability, and reliability by simulating high usage scenarios, measuring response times, and evaluating the system's ability to handle large amounts of data and users. User acceptance testing evaluates the system from the end-user's perspective. This can be done by having a group of users test the system, provide feedback, and determine if it meets their needs and requirements. Integration testing verifies that the different components of the system work together correctly. This can be done by creating test cases that involve multiple agents and evaluating the system's behavior in these scenarios. Regression testing verifies that changes to the system do not introduce new bugs or break existing functionality. This can be done by re-running functional and performance test cases after each change to the system.

The following steps for implementing the algorithm in Java using JADE and Fuzzy JESS:

(1) Create a fuzzy inference system using Fuzzy JESS to compare articles and determine their similarity based on various factors such as keywords, author, and publication date.

(2) Develop a JADE agent to represent the user. The agent should be able to send a request to the system to find similar articles to a given article. The request should contain the details of the article, such as the title, keywords, and author.

(3) Develop another JADE agent to represent the search agent. The agent should be able to retrieve articles from the database based on the request from the user agent.

(4) Develop a Fuzzy Inference Unit that uses the fuzzy inference system created in step 1 to compare the retrieved articles to the given article and determine their similarity.

(5) Develop a feedback agent that allows the user to provide feedback on the search results. The feedback should be used to update the fuzzy inference system, so that it can improve the accuracy of the results.

(6) The search agent should then return the results to the user agent, which should present them to the user in a clear and user-friendly format.

(7) Finally, test the system thoroughly to ensure that it meets the requirements and goals.

Algorithm for Finding Similar Articles to a Given Article using MAS system with Fuzzy Inference Unit and Feedback Agent

(1) Initialize the system by creating the agents, such as the search agent, relevance feedback agent, parser agent, presentation agent, user agent, Fuzzy Inference Unit, and feedback agent.

(2) The user submits a query for a specific article by using the user interface.

(3) The search agent retrieves relevant articles from the biomedical literature databases based on the user's query and applies various search algorithms and filters.

(4) The parser agent parses the results of the search, extracts relevant information, and transforms it into a format that can be easily processed by other agents.

(5) Convert each article into a numerical representation, such as a vector, by using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency).

(6) The Fuzzy Inference Unit uses fuzzy logic to process the information from the parser agent and determine the similarity between the given article and the retrieved articles. The Fuzzy Inference Unit uses a fuzzy membership function to determine the degree of similarity between the articles.

(7) The feedback agent collects feedback from the user on the relevance of the retrieved articles. This feedback can be

used to refine the search results and improve the accuracy of the Fuzzy Inference Unit's similarity calculations.

(8) The presentation agent presents the information to the user in a clear and user-friendly format, such as a list of articles ordered by similarity to the given article.

(9) The user can provide additional feedback on the relevance of the articles, and the feedback agent updates the Fuzzy Inference Unit's similarity calculations accordingly.

(10) The Fuzzy Inference Unit continues to refine the similarity calculations based on the user's feedback and returns a updated list of articles ordered by similarity to the given article.

(11) The process continues until the user is satisfied with the results, or the system reaches a maximum number of iterations.

The data sources in the simulation process can vary depending on the specific context and objectives of the study. In the case of multi-agent systems for biomedical literature retrieval, potential data sources may include:

(1) Biomedical Databases: These can include databases such as PubMed, MEDLINE, Embase, or other domain-specific repositories that store biomedical literature and related information. These databases can provide a wide range of textual data, including research articles, abstracts, keywords, author affiliations, and citation networks.

(2) Knowledge Bases: Knowledge bases such as MeSH (Medical Subject Headings) or ontologies specific to the biomedical domain can be used to provide structured data and semantic relationships between biomedical concepts. These knowledge bases can enhance the system's understanding and reasoning capabilities.

(3) User Feedback and Ratings: User feedback, ratings, or reviews can be collected from researchers, healthcare professionals, or users of the system. This feedback can provide valuable insights into the system's performance, relevance of recommendations, and user satisfaction.

Regarding the results and data of different testing types:

Functional Test: Functional testing focuses on verifying the system's compliance with functional requirements. The data involved in functional testing includes test cases, expected outputs, and actual outputs. This data is used to assess whether the system functions as intended and meets the specified requirements.

Performance Test: Performance testing aims to evaluate the system's performance under different loads and conditions. The data collected during performance testing typically includes response times, throughput, resource utilization, and system scalability. This data helps assess the system's efficiency, stability, and ability to handle varying workloads.

User Acceptance Testing: User acceptance testing involves gathering feedback from end-users to assess their satisfaction and acceptance of the system. The data in user acceptance testing includes user feedback, survey responses, and usability metrics. This data provides insights into the system's user-friendliness, ease of use, and overall user satisfaction.

Integration Test: Integration testing focuses on testing the interactions and compatibility of different system components or modules. The data involved in integration testing includes test cases, input data, and output data from integrated components. This data helps evaluate the system's ability to function seamlessly when different components are combined.

Regression Testing: Regression testing involves retesting the system to ensure that existing functionality is not adversely affected by new changes or updates. The data in regression testing includes test cases, expected outputs, and actual outputs. This data helps identify any unintended consequences or regression errors resulting from changes made to the system.

It's important to note that the specific data sources and results may vary based on the study's scope, research objectives, and the specific implementation of the multi-agent recommendation system for biomedical literature retrieval.

In order to test a multi-agent system (MAS) for retrieving relevant medical articles, the following steps need to be taken. Firstly, a dataset of 3000 articles needs to be obtained, which can consist of research articles, review articles, and case reports from different fields of medicine. Secondly, the articles need to be pre-processed through text cleaning, stemming, and vectorization, and stored in a database or knowledge base. The next step is to implement and deploy the MAS system, which includes agents for retrieving articles, computing similarity, ranking articles, and returning results. Additionally, the system should have a Fuzzy Inference Unit and a feedback agent. Once the MAS system is deployed, the input article is provided to the system, which then retrieves and ranks the articles based on their similarity to the input article. Two physicians then evaluate the top n articles returned by the system, where n is specified by the system or the user, and assess their relevance by comparing them to the input article. The feedback agent receives the evaluation results from the physicians and adjusts the parameters of the Fuzzy Inference Unit accordingly. The above steps are then repeated several times, with the parameters of the Fuzzy Inference Unit being adjusted based on the feedback received from the physicians, until the system produces satisfactory results. The performance of the system is evaluated by measuring the accuracy and recall of the results. Finally, the results of the MAS system are compared with a baseline system, such as a simple keyword-based search system or a traditional information retrieval system.

A confusion matrix as in Table 1 is a commonly used tool to evaluate the performance of a binary classification system. In this case, the system is classifying articles as relevant or not relevant to a given query. The matrix shows the count of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) classifications.

The above confusion matrix for the MAS system shows that the system has classified 1490 articles as relevant and they are actually relevant, which is considered a true positive. 17 articles were classified as relevant but they were actually not relevant, which is considered a false positive. 8 articles were classified as not relevant but they were actually relevant, which is considered a false negative. Finally, 1485 articles were classified as not relevant and they were actually not relevant, which is considered a true negative.

To evaluate the performance of the MAS system, we can use various metrics such as accuracy, precision, recall, and F1 score.

- Accuracy: (TP+TN)/(TP+TN+FP+FN)=(1490+1485)/(2000)=2975/2000=0.988
- Precision: TP/(TP+FP)=1490/(1490+17)=0.989
- Recall: TP/(TP+FN)=1490/(1490+8)=0.994
- F1 Score: 2*(Precision*Recall)/(Precision+Recall)=2*(0.989*0.994)/(0.989+0.994)=0.992

The results show that the MAS system has a high accuracy, precision, recall, and F1 score, indicating that the system is performing well in classifying articles as relevant or not

relevant to a given query. The two physicians checking the results can also provide feedback on the performance of the system and suggest any improvements that can be made.

Table 1. Confusion matrix

	Actually Relevant	**Actually Not Relevant**
Predicted Relevant	1490	17
Predicted' Not Relevant	8	1485

Table 2. The confusion matrix between the physician and model

		Model		
		Relevant	**Not Relevant**	
Physician	Relevant	1491	8	1499
	Not Relevant	7	1494	1501
		1498	1502	3000

The Confusion Matrix in Table 2 compares the predictions made by a model with the ground truth provided by a physician. The matrix has two categories: "Relevant" and "Not Relevant". The goal is to see how well the model is able to predict the physician's categorization of the data.

In the "Relevant" category, the model correctly predicted 1491 out of 1499 instances, and made 8 incorrect predictions. In the "Not Relevant" category, the model correctly predicted 1494 out of 1501 instances, and made 7 incorrect predictions.

The Kappa score can be calculated to assess the agreement between the physician and the model. The Kappa score takes into account the probability of chance agreement and provides a measure of the agreement beyond chance. A Kappa score close to 1 indicates a strong agreement between the physician and the model, while a score close to 0 indicates little agreement beyond chance. The Kappa score can be calculated using the following formula:

Kappa = (Observed agreement - Expected agreement) / (1 - Expected agreement)

where Observed agreement is the number of instances that were correctly predicted by the model and the physician, and Expected agreement is the number of instances that could be expected to be correctly predicted by chance, based on the physician's predictions and the number of instances in each category.

To calculate the kappa statistic, you need to calculate the following values:

- p_o: The proportion of the total agreement between the two evaluators, which can be calculated as (1491+1494)/3000=0.9963
- p_e: The expected proportion of agreement based on chance, which can be calculated as (1491+8)*(1491+7)/(3000*3000)+(8+1494)*(7+1494)/(3000*3000)=0.9906

Finally, the kappa statistic can be calculated as k=(p_o-p_e)/(1-p_e), which indicates the degree of agreement between the two evaluators beyond chance.

In this case, k=(0.9963-0.9906)/(1-0.9906)=0.9632, which indicates a high degree of agreement between the physician and the model, with k=1 indicating perfect agreement.

7. CONCLUSIONS

In conclusion, using a Multi-Agent System (MAS) for biomedical literature search offers many advantages over traditional single-agent systems. The integration of multiple agents and sources of information allows for a more comprehensive and accurate search result, saving time and resources for researchers and healthcare providers. Additionally, the use of MAS can also provide a more scalable, secure, and user-friendly experience for searching biomedical literature. However, the design and implementation of a MAS for this purpose require careful consideration of various technical and non-technical factors, such as data complexity and heterogeneity, communication and collaboration, scalability, performance, security, privacy, and user experience. By taking these factors into account, a MAS for biomedical literature search can greatly improve the access and utilization of information in the biomedical field.

ACKNOWLEDGMENT

Author would like to thank Tafila Technical University for the support to conduct this research during academic sabbatical year 2022/2023 in Arab Open University (AOU)-Jordan.

REFERENCES

[1] Selivanov, A., Fridman, E. (2022). PDE-Based deployment of multiagents measuring relative position to one neighbor. IEEE Control Systems Letters, 6: 2563-2568. https://doi.org/10.1109/LCSYS.2022.3169999

[2] Zhang, P., Xue, H., Gao, S., Zhang, J. (2020). Distributed adaptive consensus tracking control for multi-agent system with communication constraints. IEEE Transactions on Parallel and Distributed Systems, 32(6): 1293-1306. https://doi.org/10.1109/TPDS.2020.3048383

[3] Gómez, A., Eras, L.A.C., Aguilar, J. (2021). Multi-agent systems for the management of resources and activities in a smart classroom. IEEE Latin America Transactions, 19(9): 1511-1519.https://doi.org/10.1109/TLA.2021.9468444

[4] Aryankia, K., Selmic, R.R. (2020). Neuro-adaptive formation control and target tracking for nonlinear multi-agent systems with time-delay. IEEE Control Systems Letters, 5(3): 791-796. https://doi.org/10.1109/LCSYS.2020.3006187

[5] Wang, J., Li, Y., Duan, Z., Zeng, J. (2022). A fully distributed robust secure consensus protocol for linear multi-agent systems. IEEE Transactions on Circuits and Systems II: Express Briefs, 69(7): 3264-3268. https://doi.org/10.1109/TCSII.2022.3153698

[6] He, S., Wang, H., Yu, W. (2021). Distributed fast finite-time tracking consensus of multi-agent systems with a dynamic leader. IEEE Transactions on Circuits and Systems II: Express Briefs, 69(4): 2176-2180. https://doi.org/10.1109/TCSII.2021.3125700

[7] Rostami, M., Oussalah, M., Farrahi, V. (2022). A novel time-aware food recommender-system based on deep learning and graph clustering. IEEE Access, 10: 52508-52524. https://doi.org/10.1109/ACCESS.2022.3175317

[8] Hoai Nam, L.N. (2022). Profile aggregation-based group recommender systems: Moving from item preference profiles to deep profiles. IEEE Access, 10: 6218-6245.

https://doi.org/10.1109/ACCESS.2021.3140121
[9] Xu, B., Lin, H., Lin, Y., Ma, Y., Yang, L., Wang, J., Yang, Z. (2016). Improve biomedical information retrieval using modified learning to rank methods. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 15(6): 1797-1809. https://doi.org/10.1109/TCBB.2016.2578337
[10] Xu, B., Lin, H., Lin, Y., Ma, Y., Yang, L., Wang, J., Yang, Z. (2016). Improve biomedical information retrieval using modified learning to rank methods. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 15(6): 1797-1809. https://doi.org/10.1109/TCBB.2018.2801303
[11] Althari, G., Alsulmi, M. (2022). Exploring transformer-based learning for negation detection in biomedical texts. IEEE Access, 10: 83813-83825. https://doi.org/10.1109/ACCESS.2022.3197772
[12] Al Fayez, R.Q., Joy, M. (2017). Using linked data for integrating educational medical web databases based on BioMedical ontologies. The Computer Journal, 60(3): 369-388. https://doi.org/10.1093/comjnl/bxw096
[13] Saidi, I., Mahammed, N., Klouche, B., Bencherif, K. (2023). An overview on related searches recommendation. Ingénierie des Systèmes d'Information, 28(2): 283-289. https://doi.org/10.18280/isi.280203
[14] Ritzkal, Sutriawan, Prakoso, B.A., Fanani, A.Z., Riawan, I., Fajri, H., Basuki, R.S., Alzami, F. (2023). Word search with trending reviews on Twitter. Ingénierie des Systèmes d'Information, 28(2): 351-356. https://doi.org/10.18280/isi.280210
[15] Deore, S.P. (2023). Enriching song recommendation through facial expression using deep learning. Ingénierie des Systèmes d'Information, 28(1): 225-229. https://doi.org/10.18280/isi.280126