An Autonomous Multi-Agent System for Customized Scientific Literature Recommendation: A Tool for Researchers and Students

Abdelhakim Herrouz, Mahieddine Djoudi, Houssem Eddine Degha, Bouchra Boukanoun

Ingénierie des Systèmes d'Information
Vol. 28, No. 4, August, 2023, pp. 799-814
Journal homepage: http://iieta.org/journals/isi

An Autonomous Multi-Agent System for Customized Scientific Literature Recommendation: A Tool for Researchers and Students

Abdelhakim Herrouz[1*], Mahieddine Djoudi[2], Houssem Eddine Degha[3], Bouchra Boukanoun[3]

[1] Department of Computer Science, University of Biskra, Biskra 07000, Algeria
[2] Laboratoire TECHnologies Numériques pour l'Éducation (TECHNE), Université de Poitiers, Poitiers 86073, France
[3] Department of Computer Science, University of Ghardaia, Ghardaia 47000, Algeria

Corresponding Author Email: herrouz.abdelhakim@univ-ouargla.dz

https://doi.org/10.18280/isi.280401

Received: 19 February 2023
Revised: 12 June 2023
Accepted: 26 July 2023
Available online: 31 August 2023

Keywords:
multi-agent system, ontology, personalized information filtering, recommendation system, scientific paper, search engine, usability, user interface

ABSTRACT

The ever-accelerating growth in scientific literature presents a formidable challenge for researchers and students aiming to stay abreast of the most recent findings. Previous solutions, which include content-based, collaborative-based, and graph-based filtering recommendation systems, have their own limitations, primarily their inability to efficiently manage time-consuming search engine queries, a frequent issue for students. To address these constraints, we introduce a novel tool-a multi-agent system with an intelligent filtering mechanism. This system automates the literature search and filtering process, generating search queries independently and conducting comprehensive online searches. The system comprises autonomous agents that collectively gather and analyze data from a myriad of sources. Utilizing sophisticated techniques, the intelligent filtering mechanism leverages user preferences, interests, and contextual information. Continuous learning from user feedback allows the system to iteratively refine its recommendations, providing a personalized user experience. A user-friendly interface has been developed to streamline the configuration of the search procedure, offering users an easy way to fine-tune their preferences. Evaluations indicate that our approach delivers superior performance, significantly improving the process of scientific literature recommendation. Our tool is designed to assist researchers and students by minimizing the manual effort required in literature search and filtering, thereby ensuring efficient access to pertinent information. By automating these labor-intensive tasks, our tool enables users to keep pace with the latest scientific discoveries with increased ease.

1. INTRODUCTION

The exponential proliferation of scientific articles, facilitated by their online archiving, has emerged as a byproduct of intensifying scientific inquiry. Esteemed platforms such as ACM (Association for Computing Machinery), IEEE Xplore, Elsevier, among others, now serve as repositories for millions of scientific papers. Notably, Scopus stands out as the most extensive scholarly bibliographic database, housing 71 million articles and 1.4 billion [1].

Academic search engines have become indispensable tools for researchers and students, given their ability to access an expansive domain of scholarly literature. Among these, Google Scholar stands as a premier academic search engine, providing unrestricted access to approximately 390 million pieces [2]. Other significant databases include Microsoft Academic, with over 250 million public records. Base, an online search engine developed at Germany's Bielefeld University that indexes over 241 million documents from more than 8000 different resources [2]. Additionally, more than 190 million papers have been indexed by Semantic Scholar.

Researchers and students are heavily reliant on scientific literature for a range of tasks, from research to writing scientific articles and evaluating results. However, the escalating growth of worldwide literature data has complicated the research process. The likelihood of locating papers aligned with the users' preferences has dwindled, leading to the academic community's increased reliance on academic web search engines. For students, in particular, identifying relevant academic articles can be demanding. Search engines' algorithms, which are reliant on user keywords, often churn out an overwhelming number of potential answers. Moreover, the quality of these search results can be questionable, highlighting the importance of research impact metrics, such as the number of citations, the h-index, the g-index, and the Eigen factor scores.

Confronted with precise questions and limited time, users have turned to recommendation systems to enhance search efficiency and reduce time wastage. These systems, which choose and suggest the most relevant items based on user profiles, have found applications in various industries, including business, academia, and the scientific community. The advent of "big scholarly data" has amplified the need for intelligent suggestion methods to handle information overload and maximize the use of academic resources.

In response to this demand, this study introduces a multi-agent recommender system designed to optimize the recommendations provided by academic search engines. It employs a content-based filtering method to generate improved suggestions. User-submitted papers are used by the

system to generate search queries, conduct automated online searches, gather relevant scientific articles, and apply filters to select the most fitting ones. Cosine similarity computations are performed to gauge the congruence between the user's paper and the relevant scientific papers. Based on this measure, the top N scientific papers are recommended to the user. The system also includes an intuitive interface to customize the search process.

The remainder of the paper is structured as follows: Section 2 discusses related work in the field; Section 3 defines several central concepts; Section 4 provides a detailed description of the software; Section 5 outlines the development of the tool; Section 6 presents evaluations and results; Section 7 offers a System Usability Scale Evaluation, and the paper is concluded in Section 8.

2. RELATED WORKS

An assortment of studies has surfaced in recent years, contributing significantly towards enhancing the scholarly research terrain. One of the emerging fronts in this field is the Recommendation Systems (RS), which have gained considerable traction due to their potential for delivering pertinent information. RS employ a fusion of various technologies, fields, and methodologies, demonstrating proficiency with large datasets, a focus on textual data, and a capacity for incorporating user profiles. Four primary types of scientific article recommender systems have been widely recognized: Collaborative, Content-based, Graph-based, and Hybrid recommender systems. The succeeding discourse encompasses a discussion of selected works that embody each category.

2.1 Collaborative recommender system

Recommendations are generated by collaborative recommender systems using information on user ratings [3]. Collaborative recommender systems, which underpin paper recommendation systems, hinge on the activities of academic peers, wherein the rating of publications is contingent upon their evaluations. This methodology leverages previous researchers' assessments to refine the selection of potential research papers.

Within this context, a noteworthy method to recommend scholarly articles was developed by Lee et al. [4]. This system employed a collaborative-filtering-based technique to discern researchers' interests and deliver personalized recommendations of pertinent publications, thereby conserving time and effort traditionally spent on keyword searches or manual review of academic literature. The system's operation was bifurcated into two phases. Initially, a bag-of-words model was applied to a corpus for data collection and preprocessing, with documents represented as binary vectors. Subsequent preprocessing involved stemming, performed post the removal of stop words. The system then utilized user queries and inferred preferences to generate tailored recommendations. A lazy learning technique akin to k-Nearest Neighbors (kNN) was employed to retrieve items markedly similar to the user's previous articles. The system's functionality was validated through a dual-pronged evaluation, involving a statistical analysis and a user study. Despite achieving an accuracy of 89% in suggesting papers from the correct field, the system's focus on frequency, to the exclusion of user information and temporal data, was identified as a significant limitation.

Adding to the literature, Sakib et al. [5] proposed a recommendation procedure in 2020, which amalgamated citation context with collaborative filtering to suggest relevant scholarly papers. Upon receipt of a Paper of Interest (POI), the system initiated an extraction phase, retrieving all associated citation and reference documents. These candidate papers were represented as a matrix capturing their citation relationships. The Jaccard similarity coefficient was utilized to calculate the similarity score for each candidate paper. The final score, indicating each paper's relevance to the POI, was generated by normalizing the results of the two similarity scores. The top-N most similar papers were then presented to the user. Although the proposed approach demonstrated efficacy when compared to three baselines, it relied solely on citation relation information, overlooking potentially beneficial features such as author and journal information.

2.2 Content-based recommender system

In recommender systems, content-based filtering is a highly prevalent approach. It considers the items that users have previously indicated a preference [6]. Through a comparison of the content descriptions of the papers, this method recommends papers to users based on their prospective interests.

In 2015, Hanyurwimfura et al. [7] proposed a method for recommending academic research papers to researchers without relying on user profiles. Traditional user profile-based systems necessitate user registration and recommend papers based on profile similarity. This approach has limitations, such as its incapacity to accommodate non-registered or new users who have only read a single paper and are looking for similar ones. To address this issue, the authors presented a method for recommending relevant papers based on a target paper's topic and primary ideas by generating queries from the entire paper's content. Their methodology entailed analyzing sections of the target paper that adequately characterize its primary content and provide information about related papers. They concentrated on the target and candidate papers' titles, abstracts, introductions, and related work sections. Four algorithms for generating topics for brief or long queries have been proposed. The first algorithm utilized the target paper's title and citations to generate brief queries. The second algorithm extracted the paper's main idea from the abstract to generate lengthy queries by using cue words. The last two algorithms selected only a few sentences or phrases from the body of the paper that was highly germane to its main ideas, generating either long or short queries. The generated query was then submitted to an online repository to retrieve candidate papers, from which the most relevant were chosen for recommendation. Using cosine similarity, the authors measured the similarity between specified fields of the target paper and candidate papers. The paper was recommended if cosine similarity determined that the content was similar. Using Recall and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics for various query generation and ranking strategies, the accuracy of their method was measured. Experiments were conducted to evaluate the proposed paper recommendation methods, which revealed a significant improvement in results.

The authors of "A Content-Based Approach to Citation Recommendation in Academic Paper Drafts" by Bhagavatula

et al. [8] presented a method for recommending citations that can improve the quality and efficacy of the literature review process. Instead of relying on metadata such as author names, they use a neural model to encode the textual content of all available documents and embed them into a vector space. To accomplish this, they used Sentence-BERT (SBERT), a transformer-based text embedding technique. Using positive and negative examples from the citation graph, the authors fine-tuned SBERT. During the prediction phase, they also proposed a submodular scoring function to reconcile the relevance of recommended citations with the diversity of their authors. The authors evaluated their recommendations using precision, recall, Mean Reciprocal Rank (MRR), and the F1@k score. MRR determines the position of the first correct citation in the recommended list by calculating its reciprocal and aggregating it across the test set. The F1@k score indicates the harmonic mean of precision and recall at corpus position k. Precision and recall were initially calculated for each inquiry document before being averaged across the test set to determine the F1 score. Across all metrics, the experimental results conducted on the ACL Anthology Network corpus, a benchmark dataset, demonstrated that the proposed method is superior to other methods, including a state-of-the-art neural method. In addition, the authors demonstrated empirically that while the incorporation of metadata enhances the performance of standard metrics, it tends to favor self-citations, which are less valuable in the context of citation recommendation.

2.3 Graph-based filtering

Graph-based methodologies are increasingly becoming indispensable to the analysis of diverse real-world systems. Nevertheless, comprehensive scrutiny of these strategies remains scant.

In 2017, Dai et al. [9] proposed TMALCCite, an innovative citation recommendation approach for bibliographic networks, which leverages a novel topic model. The TMALCCite model enriches the recommendation process by amalgamating two key elements: the textual content similarity among research papers and the community relevance among authors. This approach extends the Latent Dirichlet Allocation (LDA), a statistical model, by incorporating the intricate relationship between textual content similarity and community relevance, thereby enabling practical and robust recommendations. By integrating semantic and link information of the research papers, TMALCCite facilitates the concurrent learning of lower dimension spaces for paper nodes, author communities, and topics. A parameter inference algorithm, grounded in Maximum A posteriori (MAP) estimation, was developed, with ample evidence of algorithm convergence provided by Dai et al. [9]. The authors further enhanced the model's efficiency by introducing a range of citation link probability functions, which have been shown to bolster recommendation performance. To assess the efficacy of their model, the authors juxtaposed TMALCCite against several existing algorithms, including BM25, PopRank, Random walks, ClusCite, LDA, Link-PLSA-LDA, TLLDA, and RTM. Trials conducted on the AAN and DBLP datasets, using citation information from training papers to train the model and citation information from test papers as the ground truth, demonstrated TMALCCite's superiority across all evaluation metrics, including precision, recall, and mean reciprocal ranking (MRR).

In a related work, Ma and Wang [10] unveiled HGRec, a novel method for paper recommendations that employs a heterogeneous graph representation to address the complexities of personalized paper recommendation. A heterogeneous graph, a directed graph characterized by diverse node and link types, was employed in their method. The authors formulated user and paper profiles based on extracted content information from the research papers, which included titles, keywords, and abstracts. These profiles were then utilized to generate initial embeddings for users and papers using the pre-trained Doc2vec technique. The HGRec method proposed by Ma and Wang [10] permitted the learning and updating of embeddings for various node types within the heterogeneous graph. The final list of recommended papers for the target researchers was obtained by calculating the cosine similarity between the final user feature vectors and paper feature vectors. The HGRec method was subjected to a rigorous evaluation, which involved a comparative study against state-of-the-art methods such as Content-Based Paper Recommendation (CBR), Graph-Based Paper Recommendation (GBR), MPRec, HGRec, and HGRec1. The evaluation metrics employed included precision, recall, and F-measure. The results demonstrated a notable superiority of the HGRec method over the other approaches. However, the authors acknowledged a limitation in their approach: the descriptive paths utilized in their method were manually created.

2.4 Hybrid recommender system

Hybrid recommendation methodologies, built upon the integration of multiple techniques, have proven instrumental in the development of high-performing scientific paper recommender systems. By amalgamating content-based and collaborative-based approaches, these systems are able to offset their individual shortcomings while capitalizing on their strengths.

In 2015, Meilian et al. [11] proposed a method for recommending academic papers of varying quality levels within network-based systems. Their approach involved the use of the Advanced Hyperlink Induced Topic Search (AHITS) algorithm, designed to recommend high-quality academic papers to users, thereby expanding their academic purview. The AHITS algorithm utilizes a tripartite graph, termed UPT (User-Paper-Topic), to assess the quality and authority of academic resources. While the experiment yielded promising results in addressing the identified challenge, their method was shown to still be susceptible to the cold start problem.

In a subsequent study conducted in 2020, Shi et al. [12] introduced an AMHG-based hybrid paper recommendation method, predicated on a multi-level citation heterogeneous graph. In an effort to alleviate the cold start problem, the authors not only considered similar papers published by the same author but also incorporated metadata of the papers into their model. The model was further refined by reordering the output candidate list based on the author influence factor, with the intention to prioritize high-quality papers. The investigation employed the DBLP-REC dataset and the results demonstrated a significant improvement in the accuracy of recommendations provided by the AMHG method. However, this approach was found to rely exclusively on offline data, thereby limiting its capacity to provide personalized recommendations. This limitation was attributed to the lack of a user's usage record and the exclusion of key metadata such

as the publication journal.

It is evident from the literature review that significant efforts and research have been dedicated to the development of filtering techniques for recommender systems. Despite these advancements, the reviewed methods exhibit inherent limitations. Collaborative filtering recommender systems necessitate vital information about users and items prior to generating recommendations. However, in instances where there is no information about the user or the item in the system, or when only a few ratings from users about various items are available, numerous challenges, including the cold start and sparsity problem, arise. The conceptual modeling of our proposed paper recommendation system, designed to address these issues, will be elaborated upon in the following section.

3. CONCEPTS DEFINITIONS

In this section, we will present the definitions of the most important concepts and theorems used in our research to develop the new tool.

3.1 Bayes theorem definition

Thomas Bayes formulated a mathematical theorem to calculate conditional probability, which is the probability of an event occurring based on a prior event. Bayes' theorem takes into account prior probability distributions to generate posterior probabilities. Prior probability is the probability of an event before new data is collected, and it represents the best rational estimate of the probability of an outcome before an experiment is performed. The posterior probability is the updated probability of an event occurring after new information is considered, and it is calculated using Bayes' theorem. In statistical terms, the posterior probability is the probability of event A occurring, given that event B has occurred. If A and B are two events in a sample space S, then the conditional probability of A given B is defined as:

$$P(A \vee B) = \frac{P(A \cap B)}{P(B)}, \text{when } P(B) > 0. \quad (1)$$

where,

P(A)=The probability of A occurring

P(B)=The probability of B occurring

P(A|B)=The probability of A given B

P(A∩ B)=The probability of both A and B occurring, where is the joint probability of both A and B being true.

Because,

P(B∩A)=P(A∩B)⇒P(A∩B)=P(AB)P(B)=P(BA)P(A)

The Bayes' Rule, for any two events A and B, where P(B) ≠ 0, we have

$$P(A \vee B) = \frac{P(B \vee A).P(A)}{P(B)} \quad (2)$$

3.2 Markov chains

A Markov chain is a stochastic model that describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. If the chain moves state at discrete time steps, it is called a discrete-time Markov chain (DTMC).

Consider the random process $\{X_n, n=0,1, 2,\}$, where $RX_i=S \subset \{0,1, 2...\}$. We say that this process is a Markov chain if:

$$P(X_{m+1}=j|X_m=i_{m-1}=i_{m-1}, ...,X_0=i_0)=P(X_{m+1}=j|X_m=i) \quad (3)$$

for all m, j, i, i_0, i_1, ..., i_{m-1}. If the number of states is finite, such as S = {0, 1, 2, ..., r}, we call it a finite Markov chain.

4. SOFTWARE DESCRIPTION

This section introduces the MASSPR (Multi-Agents System for Scientific Paper Recommendation). The MASSPR tool consists of six primary modules: Graphic User Interface (GUI), Front-end modules, GUI Back-end Engine, Query Module, Searching Module, Data preparation and learning module, and Recommendation and Filtering Module. Each module performs specific tasks and collaborates with others to achieve the system's objectives.

What sets MASSPR apart from other recommendation systems is its unique approach of extracting queries from a research paper provided by the user instead of relying on user-provided keywords to represent their interests. These queries, derived from the most relevant terms in the paper, are then submitted to existing online repositories storing academic papers. The system retrieves similar papers from these repositories to generate recommendations.

To accomplish its recommendation goals, MASSPR employs a multi-agent system with an intelligent filtering mechanism. It suggests all relevant papers that users may find valuable based on their input paper. Figure 1 depicts the architecture of MASSPR, providing an overview of the system's structure.

4.1 Query modelling module

The Query Modeling Module is a sub-module of MASSPR that plays a crucial role in constructing and modeling searching queries. Figure 2 provides an insight into the internal architecture of this module. One of the significant challenges students face when using search engines is the inadequacy of their keywords and queries in finding papers relevant to their needs. Our tool addresses this issue by generating search queries based on a series of base papers the user provides.

This modeling approach utilizes a selection mechanism grounded in the Bayes theorem to extract concepts and keywords. The process begins by breaking down the user-provided papers into distinct sections: title, abstract, author and affiliation, keywords, introduction, references, and more. Each section undergoes preprocessing to transform the text into a more manageable format, thereby enhancing the performance of this module. During the preprocessing phase, various filters are applied, including removing special characters and white spaces, converting texts to lowercase, transforming number words into numeric form, and other necessary transformations. Subsequently, similar sections are grouped into bags. The module calculates the conditional probability of their co-occurrence using Bayes Theorem for each pair of consecutive words within each bag. This step generates a list of conditional probability values for each section, which are then stored in a temporary database. Before the search queries are modeled based on the previous results, the calculated probabilities are sorted and arranged in descending order to select the highest probability.

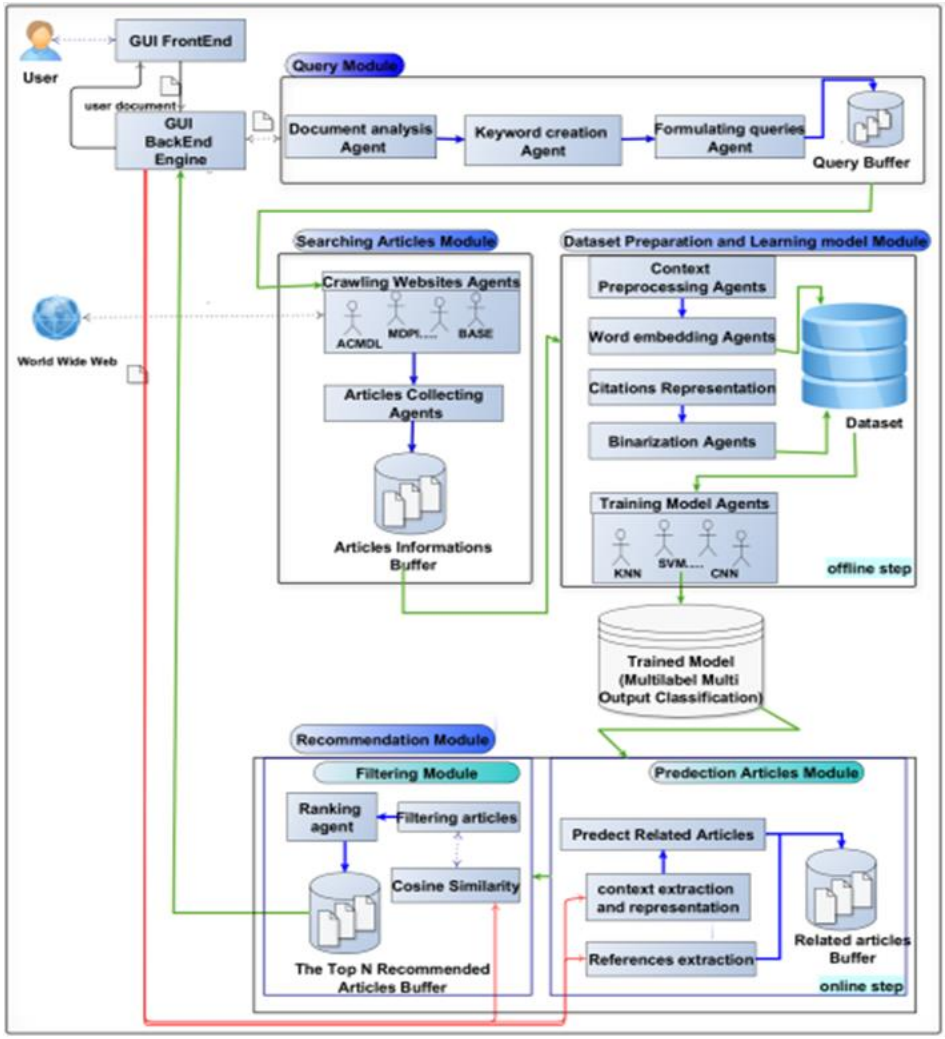

Figure 1. The general architecture of the MASSPR system

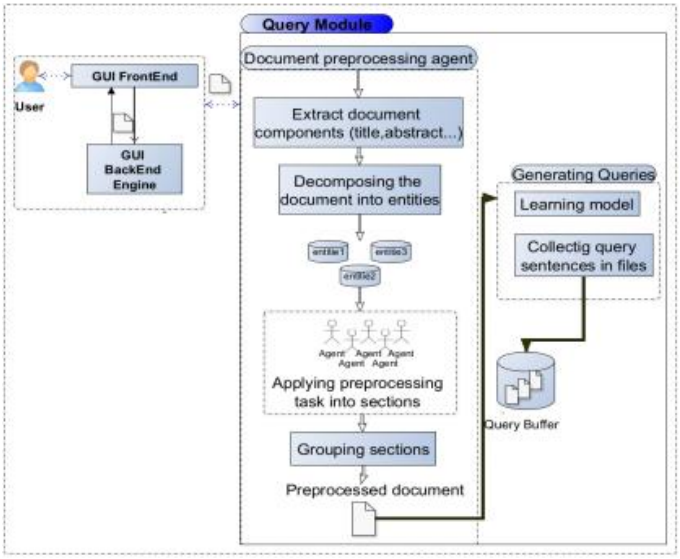

Figure 2. Query modelling module sub-architecture

Overall, the Query Modeling Module within MASSPR is designed to address the issue of ineffective keyword and query usage by generating search queries based on a series of base papers. By utilizing a selection mechanism rooted in the Bayes theorem and applying preprocessing techniques, the module enhances the quality of the generated queries, leading to more accurate and relevant recommendations.

The query modeling phase employs the Markov Chain technique to construct coherent sentences for searching queries based on a list of consecutive words. This process begins by selecting the top consecutive words from the temporary database. For each pair of words, the second word is combined with other consecutive words that follow it and possess the highest probability. This technique generates sentences that effectively represent the content of the given papers. The process above is repeated for each consecutive word, resulting in a list of sentences. Finally, the probability of each sentence is calculated using a formula derived from the Markov chain theorem. Using the Markov Chain technique, the query modeling phase generates well-formed sentences for search queries. This approach enhances the effectiveness of the MASSPR system by ensuring the relevance and coherence of the queries, ultimately leading to more accurate recommendations.

The most significant values will be chosen to accurately represent the given papers. Subsequently, the modeled sentences will be transformed into URLs format and stored in the query buffer submodule. This conversion facilitates the collection of scientific papers from various publisher web portals, including Elsevier, Springer, Google Scholar, and others. Figure 3 exemplifies the probability computation for a modeled sentence, showcasing the functionality of the Query Modeling Module within MASSPR.

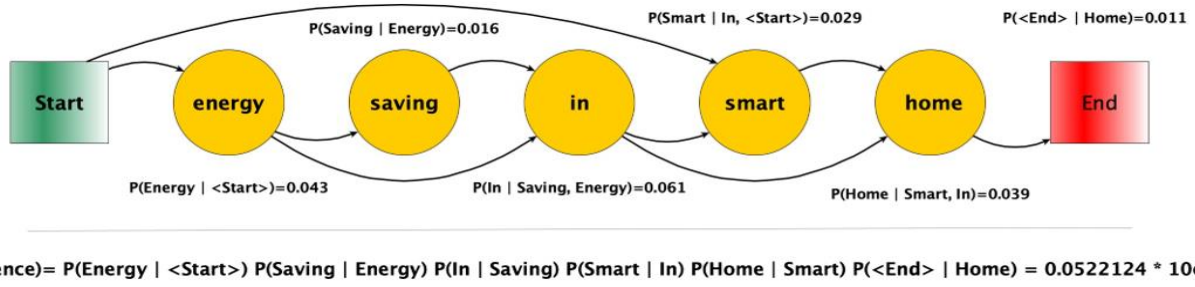

Figure 3. Example of a modeled sentence using Markov Chain technique

$$Sentence = \sum_{i=0}^{m} Word_i$$

$$P(Sentence) = P(\ Wor\,d_m) * \prod_{i=m-1}^{1} \frac{P(\ \text{Wor}\,d_{i+1} \vee \text{Wor}\,d_i).P(\ \text{Word}_i)}{P(\ \text{Wor}\,d_{i+1})} * P(\ Wor\,d_0), m \geq 3 \quad (4)$$

4.2 Documents preprocessing agent

This agent plays a crucial role in text preprocessing, which involves cleaning the input document's textual data to prepare it for building a machine learning model. The document preprocessing phase encompasses various techniques to process textual data effectively. The process begins by extracting the different components of a given paper provided by the user and dividing them into individual parts. Each section represents a specific part of the paper, such as the title, abstract, keywords, introduction, references, etc. These sections then undergo the preprocessing phase to transform the textual content into a more manageable format, enhancing the model's performance. During the preprocessing phase, several filters are applied to the sections. These filters include removing special characters, spaces, and converting the text to lowercase. Additionally, numeric words are converted to their numeric form. Other preprocessing techniques may also be utilized to ensure the data is in a suitable format for further analysis. Finally, the preprocessed sections that share similarities are grouped together to form a complete document. This process consolidates the individual parts into a cohesive whole, ready to be utilized for building the machine learning model. The input document is cleaned and transformed into a more digestible form through the text preprocessing performed by this agent. The resulting data is better prepared for subsequent analysis and modeling tasks by applying various techniques and filters.

4.3 Query generation agent

The preprocessed document, obtained from the documents pre-processing agent, is utilized as input for a machine learning-based model. This model automatically incorporates a transformer architecture to extract the most relevant keywords from the document. Within seconds, the model identifies keywords and phrases that describe the document's content.

Once the important words and phrases are extracted, the model constructs search query sentences using these significant terms. This step enables the model to generate concise and effective search queries that capture the essence of the document. Subsequently, the agent responsible for this process collects the generated search query sentences and stores them in files, specifically within a query buffer. This storage mechanism ensures that the search queries are readily available for subsequent stages of the recommendation system. The system efficiently extracts essential keywords from the preprocessed document by employing a machine learning model with a transformer architecture. This approach streamlines constructing search queries, allowing for more accurate and meaningful recommendations. Storing these queries in the query buffer ensures their accessibility and usability within the recommendation system.

4.4 Searching articles module

The Searching Articles Module is tasked with searching for scientific papers on the web using the generated queries. It comprises three main components.

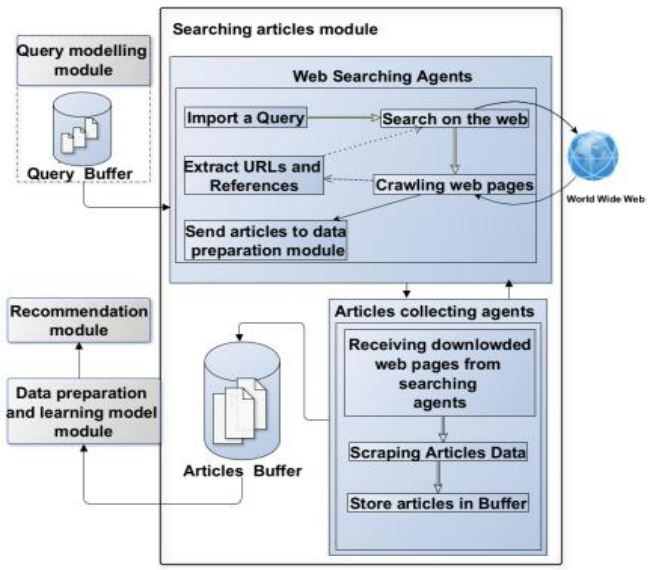

Figure 4. Sub-architecture of the searching articles module

First, the web searching agents, functioning as spiders, utilize the queries to search the web for relevant scientific papers. The inner architecture of the searching agent is depicted in Figure 4. The process begins by importing queries from the query buffer. The agent then performs a search task, crawling web pages and extracting URLs that are deemed relevant. Each discovered article is subsequently forwarded to the collecting agents in the next module. To facilitate web navigation, we leverage the Jsoup Java library, which provides a user-friendly API for fetching URLs, extracting data, and manipulating HTML using HTML5 DOM methods and CSS selectors. Second, the article collecting agents are responsible for aggregating the downloaded articles obtained from the

previous module. These agents scrape articles and extract relevant data stored in a well-structured and unified format within the articles buffer. Finally, the articles buffer acts as a temporary database that houses the gathered scientific papers from the web. The agents within our search module are autonomous, meaning they can make judgments regarding the collected material. These judgments are based on the Bayes probabilities computed by the preceding model.

Figure 5 presents the sub-architecture of the searching articles module, providing an overview of its internal structure and component interactions. Through the collaborative efforts of the web searching agents, article collecting agents, and the articles buffer, the Searching Articles Module effectively retrieves scientific papers from the web and stores them in a structured manner. The agents' autonomy allows for intelligent decision-making, while utilizing the Jsoup library streamlines the web navigation process.

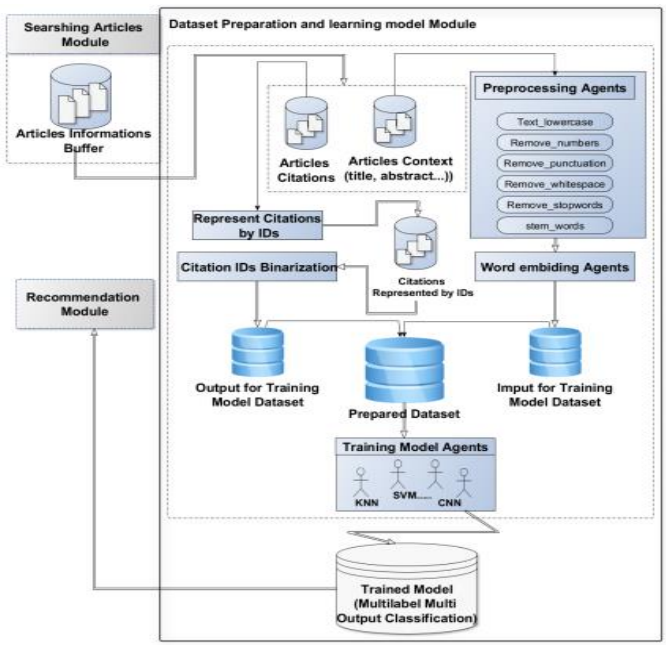

Figure 5. Data preparation and learning module sub-architecture

4.4.1 Web searching agents

The Searching Articles Module consists of a team of agents that function as web spiders, utilizing the queries to search the web for relevant scientific papers. These agents begin their process by importing queries from a query buffer, and then initiate search tasks across the World Wide Web. They navigate through various websites, automatically retrieving targeted web pages. The agents extract relevant URLs from each webpage during their web crawling process. These URLs serve as potential sources for finding scientific papers. The agents explore these URLs, downloading the associated web pages for further analysis. Once the web pages are downloaded, the agents transfer them to the collecting agents in the subsequent module for further processing and aggregation.

The collective efforts of these agents enable the Searching Articles Module to effectively search the web, retrieve relevant scientific papers, and prepare them for subsequent stages of the recommendation system.

4.4.2 Articles collecting agents

Upon receiving the downloaded web pages from the web searching agents, the next crucial step is to scrape the articles and extract relevant data. The agents responsible for this task selectively store the well-structured scientific papers obtained from the web in the article buffer, ensuring their availability for future use in subsequent modules. To optimize the data extraction process, our approach focuses on targeting specific sections of the documents that contain essential information, while disregarding sections that are not directly relevant to the primary contribution of the papers. By doing so, we aim to extract the most pertinent details from the articles. In particular, we concentrate on scraping sections that adequately describe the main content of the papers. This includes extracting information such as the title, abstract, and keywords, which play a vital role in understanding the core aspects of scientific work. By selectively extracting and storing the relevant sections of the articles in a well-structured manner, the agents ensure that the essential information is readily accessible within the article buffer. This curated collection of scientific papers is a valuable resource for subsequent modules within the recommendation system.

4.5 Data preparation and learning module

4.5.1 Learning model agent

In our system, we employ a multi-label classification learning model to predict the citations of an article based on its context. In multi-label classification, the model's predictions consist of a collection of labels for each instance. In our case, the input for the classification model is the article's context, which encompasses information such as the title, abstract, keywords, and authors. The model's output comprises a set of predicted citations or references for the article. For example, given a specific context, the model can classify the citations as citation-1, citation-2, and so on (Figure 5).

4.5.2 Data preparation

They must be prepared before feeding the articles from the buffer database into the learning model. The data for the learning model is divided into two parts: the inputs and the outputs for the classification model. Both of these parts require processing to meet the requirements of the multi-label classification model.

Reference Extraction and preparation (output model): The references from all citing documents are extracted and represented by IDs corresponding to the ranking of the papers in the buffer. Subsequently, a binarization transformation is applied to prepare the output for the multi-label, multi-output classification model. During the learning stage, this binarization transformation involves training a regressor or binary classifier for each class. It requires converting the multi-class labels into binary labels, indicating whether an instance belongs or does not belong to a particular class. The scikit-learn library's LabelBinarizer provides a convenient transform method for performing this conversion.

The first step is to extract the contexts of all articles from each citing document. These citation contexts are then transformed into the desired representation format, involving preprocessing and embedding vector transformation. Like other NLP problems, the input text data must undergo preprocessing before being fed into the model. The preprocessing stage includes applying filters to remove special characters and white spaces, converting text to lowercase, number words to numeric form, and other relevant transformations.

4.6 Recommendation module

The architecture of the Recommendation Module system is depicted in Figure 6 and comprises two main components: the Model Prediction Module and the Filtering Articles Module.

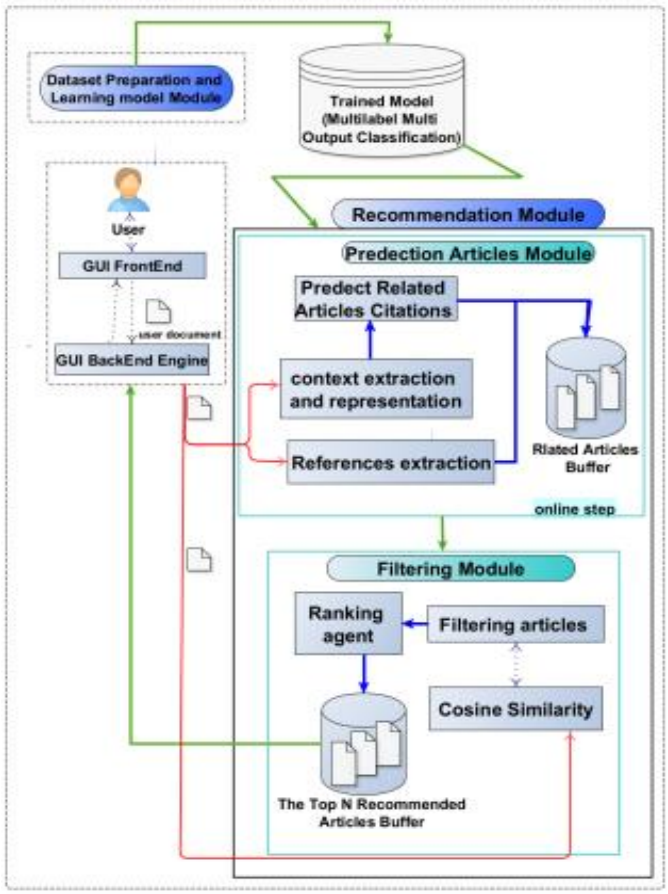

Figure 6. Recommendation module sub-architecture

4.6.1 Model prediction module

The Model Prediction Module of the Recommendation Module system begins by collecting and saving all the references mentioned in the user's input article. It then extracts and represents the contextual information of the user's input article using the same methodology employed in the offline step. This involves preprocessing the data and generating embedding vectors. Subsequently, the trained model is applied to the represented user article context to generate predictions for citations, which serve as the outcome of the model prediction.

4.6.2 Filtering article module

The Filtering Articles Module plays a crucial role in selecting a subset of potentially relevant articles to the user's needs. The filtering agents collaborate to choose the most representative articles based on the user's search query. The underlying concept of the filtering algorithm relies on intelligent machines developed using Bayes and Markov chain theorems. The filtering process begins by constructing a graph of nodes representing words and articles. These graph nodes are interconnected through semantic relationships. Each word is associated with three types of semantic relations. Firstly, PW, which represents the Probability of a Word Appearing in a Scientific Paper. Secondly, Pwc, measures the Probability of a Word Appearing in a Cited Paper. And thirdly, Pwa denotes the Probability of a Word Appearing in other authors' Papers. These three probabilities are utilized in the following formula to compute the power of the word node:

- The Formula to compute the words node power in the Masspr graph.

$$Word = \sum_{i=0}^{m} Pw_i + \sum_{j=0}^{n} Pwc_j + \sum_{k=0}^{h} Pwa_k$$

The power of the word node plays a crucial role in determining the power of the article node. The article node incorporates several additional relationships from the three semantic relationships discussed earlier. One of these relationships is the CBP, which indicates that Paper 1 cites Paper 2. Additionally, we have the PMA, which measures the Probability of Mutual Authors, and the CSP, representing the Probability of both papers cited in the same paper. To calculate the power of the article node, the following formulas are employed:

- The Formula to compute the Article node power in the MASSPR graph

$$Articlei\ r0 = \sum_{i=0}^{m} Cb\ pi + \sum_{j=0}^{n} Pm\ aj + \sum_{k=0}^{h} Cs\ pk + \sum_{z=0}^{f} Wordi\ r0$$

$$Articlei\ rid = \frac{P(\text{Articlei r id} \cap \text{Articlei r id-1})}{P(\text{Wor } d_{i+1})} * \text{i}$$
$$(\sum_{i=0}^{m} Cb\ p\ id\ i + \sum_{j=0}^{n} Pm\ a\ id\ j + \sum_{k=0}^{h} Cs\ p\ id\ k + \sum_{z=0}^{f} Word\ i\ r\ id\ i), i > 0$$

Figure 7 showcases an example of a MASSPR graph that incorporates words, articles, and various semantic relationships. The Agents traverse the graph nodes in each iteration using our innovative mechanism. They begin with the article possessing the highest power and then proceed to the following article ranked second. The agents compute the power of the second article by considering the conditional probability that assumes its existence relies on its predecessor. This computation utilizes the Markov chain, which only considers one predecessor for calculating the conditional probabilities of occurrence. Additionally, we leverage the Bayes theorem to ensure the computation reflects the intended logic. This mechanism generates a dynamic phenomenon where the power of articles and words changes, with some decreasing sequentially. During each iteration, nodes whose power falls below a certain threshold, known as the Learning Rate, are deleted. By following our algorithms, node powers decrease with each iteration, reducing the number of articles until we obtain the desired number specified by the user. We employ the concept of Entropy to assess the amount of useful information contained in the article set. With each iteration, the Entropy increases until it reaches its maximum value with the minimum number of articles. This signifies that the remaining articles can effectively represent the entire set and are the most relevant ones that fulfill the users' needs.

This research paper employs the cosine similarity function to measure similarity within each field. The cosine similarity function is a well-established measure that provides accurate results. It quantifies the cosine angle between two vectors, serving as a reliable metric for assessing similarity. This function is commonly utilized in various domains, including information retrieval and text mining, to compare and evaluate text documents [10].

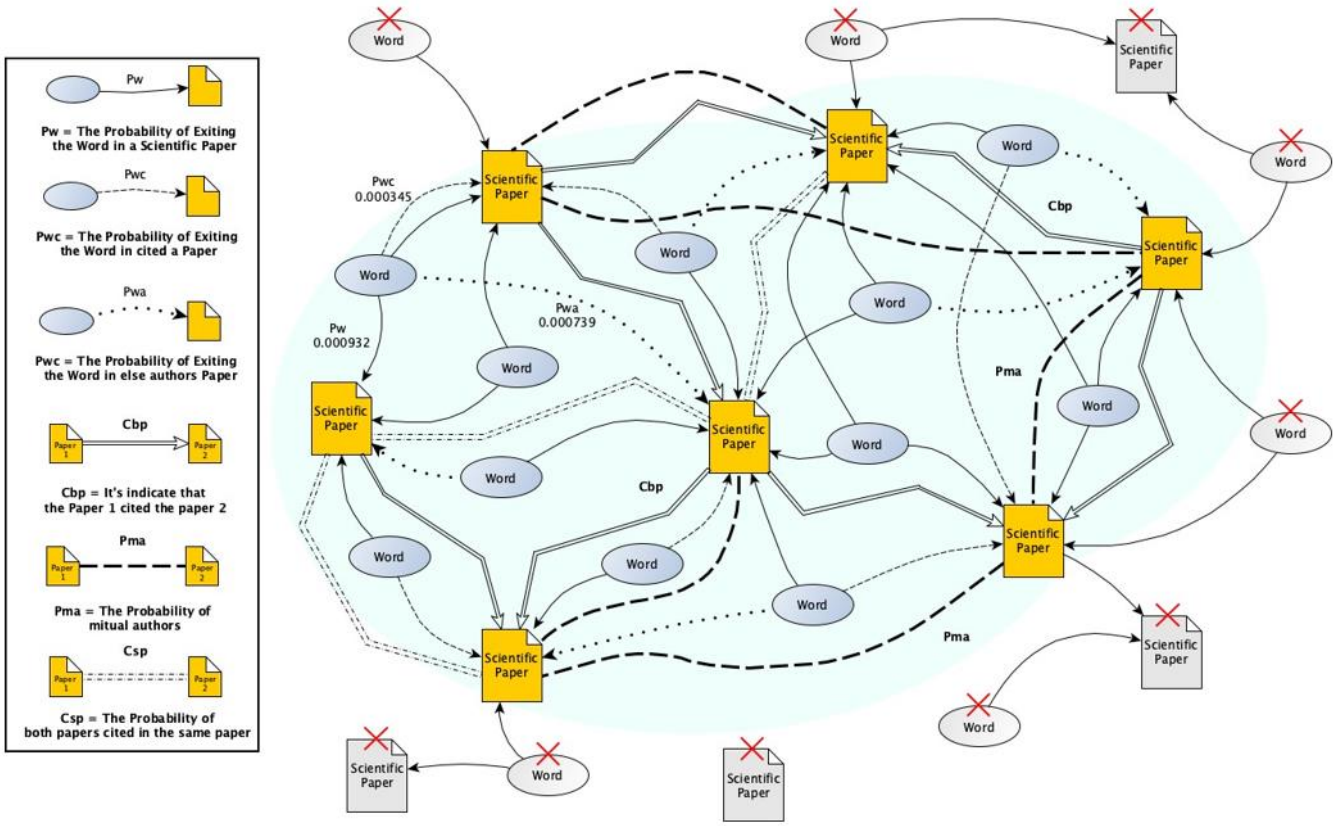

Figure 7. An example of a builder graph of concepts and articles by filtering agents

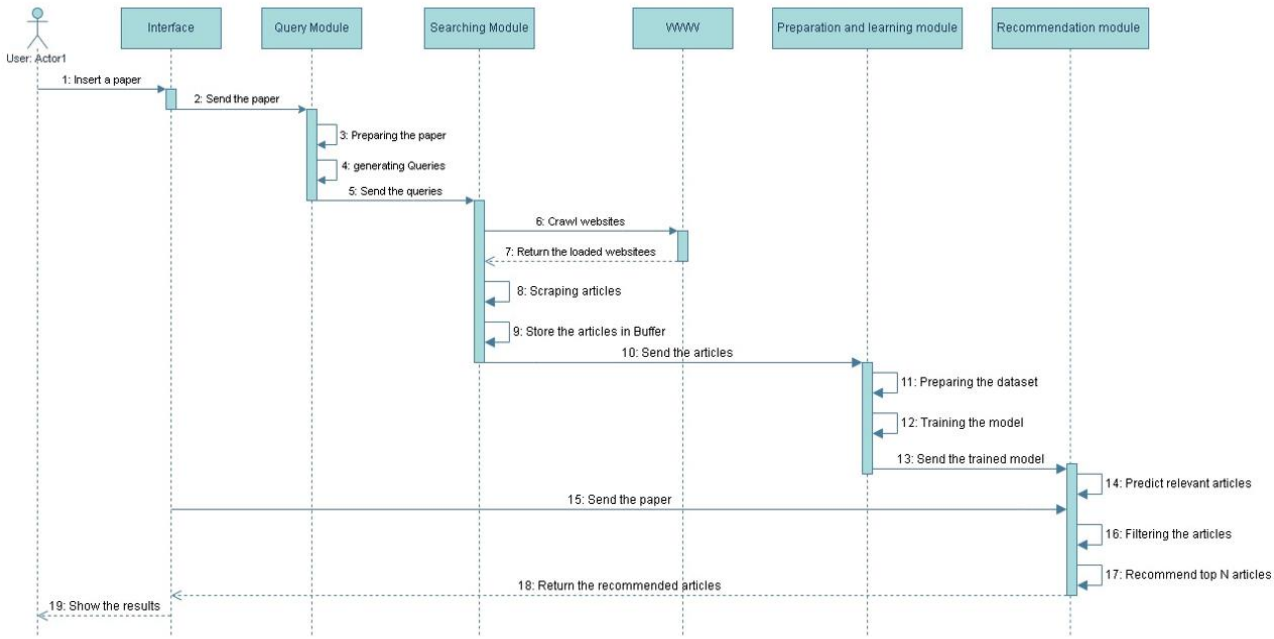

Figure 8. General sequence diagram

4.7 GUI front-end module and GUI back-end engine

The GUI Front-End module serves as the user interface for interacting with the system, while the GUI Back-End Engine module plays a crucial role in providing the necessary functionalities to support the GUI Front-End module. Acting as a bridge between different modules in MASSPR, such as the Query Module and Filtering Module, the GUI Back-End Engine module performs the following roles:

- Execution of visualization functions: The GUI Back-End Engine module handles the execution of all visualization functions required for the smooth operation of the GUI Front-End module.
- Integration with the Query Module: This module facilitates the transfer of scientific articles uploaded by the user to the Query Module, enabling the generation of search queries based on the user's input.
- Facilitation of recommended papers: The GUI Back-End Engine module takes the responsibility of sending the

recommended scientific papers, which have been selected by the filtering model, to the GUI Front-End module. These papers are then displayed to the user for further exploration and analysis.

4.8 Sequence diagram

The sequence diagram illustrates the interactions and flow of events between components and objects within the system (Figure 8). Here is a revised version of the Sequence Diagram Description:

(1) The user initiates the system by inputting the desired paper into the interface to find similar papers.

(2) The interface transmits the user's paper to both the query and recommendation modules.

(3) The query module performs a preprocessing task on the paper to facilitate interpretation. It identifies suitable keywords and constructs search queries based on these keywords.

(4) The query module sends the constructed queries to the searching module.

(5) The searching module utilizes the queries received from the query module to crawl websites on the World Wide Web and scrape relevant articles. The retrieved articles are stored in a buffer.

(6) The data preparation and learning module retrieves articles from the search articles buffer and prepares them for training the learning model.

(7) The trained model is then sent from the data preparation and learning module to the recommendation module.

(8) The recommendation module utilizes the trained model to predict relevant articles based on the user's paper. It filters these relevant articles by calculating the cosine similarity between each article and the user's paper.

(9) The recommendation module selects the top N articles with the highest similarity scores.

(10) The selected articles are returned to the interface, where they are displayed to the user as the recommended similar papers.

5. DEVELOPMENT

Our objective is to enhance a system that assists students and researchers in automatically collecting and filtering published scientific papers from the Internet, while recommending the most pertinent ones within their research domains. To achieve this, we outline the following goals of MASSPR, which aim to empower users to:

(1) Perform document-based search using input documents: Users can input their desired documents into the system, and the system will utilize these documents to conduct searches for related scientific papers.

(2) Choose the scientific platform for the search: Users can choose the scientific platform or database where the search will be conducted, allowing them to tailor the search to their specific needs.

(3) Provide the most pertinent document as the output: The system employs advanced algorithms and machine learning techniques to analyze and rank the retrieved scientific papers, ensuring that the most relevant and significant documents are presented to the users.

(4) Present the details of the retrieved related documents: Along with the recommendation of pertinent scientific papers, the system provides comprehensive details about each retrieved document. This includes information such as the title, abstract, author, keywords, and other relevant details that aid users in assessing the relevance and significance of the papers.

By achieving these goals, MASSPR empowers users to efficiently search for and access valuable scientific papers, saving them time and effort in their research endeavors.

To create the MASSPR tool, we utilized the Python programming language and the Jupyter IDE. We also employed the Java programming language and the NetBeans IDE as the development environment.

5.1 Transfer learning

Transformer-based pre-trained language models have demonstrated remarkable achievements across various natural language processing (NLP) tasks. The development of these models commenced with GPT and BERT, both of which rely on transformers, self-supervised learning, and transfer learning. By leveraging the power of transformers, pre-trained language models acquire universal language representations through self-supervised learning, and subsequently transfer this knowledge to downstream tasks. This approach offers the advantage of furnishing downstream models with valuable foundational knowledge, obviating the need to train them from scratch [10].

(1) doc2query/all-t5-base-v1

The T5-base model underwent training using the MS MARCO Passage Dataset, comprising approximately 500,000 actual search queries sourced from Bing and their corresponding relevant passages. This model serves the purpose of query generation, enabling the acquisition of semantic search models without the need for annotated training data. This technique is known as Synthetic Query Generation. Another model, called doc2query or docT5query, builds upon the T5 model. It can be employed for document expansion by generating 20-40 paragraph queries. These paragraphs and generated queries are then indexed using a standard BM25 index such as Elasticsearch, OpenSearch, or Lucene. One notable advantage of the doc2query approach lies in its simplicity in terms of conceptual understanding and implementation. The model can be easily trained using existing sequence-to-sequence neural toolkits with minimal modifications required [12].

(2) all-MiniLM-L6-v2

Table 1. All-MiniLM-L6-v2 model information

Description	All-Round Model Tuned for Many Use-Cases Trained on a Large and Diverse Dataset of Over 1 Billion Training Pairs
Base Model	nreimers/MiniLM-1.6-H384-uncased
Max Sequence Length	256
Dimensions	384
Normalized Embeddings	true
Suitable Score Functions	Dot-product (util dot_score), Cosine-similarity (util.cos sim), Euclidean distance
Size	NO MB
Peoling	Mean Pooling
Training Data	18+ training pairs for details see model card
Model Card	https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

The Sentence Transformers Library 2 is a publicly available collection of advanced sentence encoder models that are based on the architecture of Sentence-BERT. This library includes various state-of-the-art models capable of transforming text into vector representations. Sentence-BERT utilizes the BERT encoder within a Siamese architecture, training it for similarity comparison tasks. While Sentence-BERT can be trained on data from different domains to cater to various tasks, there isn't a specific encoder within the library that is specifically trained for encoding Java code. However, the generic pre-trained all-MiniLM-L6-v2 encoder is available. This encoder is trained on a diverse, extensive dataset of training pairs to support multiple domains [13].

Table 1 contains details about the all-MiniLM-L6-v2 model.

5.2 GUI front-end and GUI back-end module

To commence the development of the MASSPR system, we prioritize the construction of its core, which encompasses all essential modules and their respective functions. The Graphical User Interface (GUI) Front-End module serves as the platform for user interaction, and we leverage the Java programming language along with JavaFX libraries to create a user-friendly GUI that caters to the specific needs of our users. Our tool is designed for desktop software usage, providing a seamless experience. The MASSPR interface offers a range of notable features, a few of which are highlighted below:

- Time-saving capabilities: Users can streamline their search tasks by utilizing customizable web search agents tailored to their specific requirements. This feature ensures efficient and targeted searches, saving valuable time.
- Query Module: The Query Module facilitates uploading a collection of papers and generates precise search queries based on the uploaded content. This allows users to obtain relevant results quickly and accurately.
- Easy access to recommended papers: Users can conveniently browse through recommended scientific papers and export them to their local machine repository. This feature enables easy access to important research materials.

The GUI Back-End Engine module in MASSPR is the central hub for various internal modules, such as the query and recommendation modules. It plays multiple roles, including executing visualization functions essential for the proper functioning the GUI Front-End module. Additionally, it sends the scientific articles uploaded by the user to the Query Module for query modeling. It transmits the recommended scientific papers, filtered by the model, to the GUI Front-End module for display.

5.3 Query module

The query module leverages the power of the doc2query/all-t5-base-v1 model to generate queries. These queries play a crucial role in bridging the lexical gap in lexical search by incorporating synonyms. Moreover, the model employs word reweighting techniques to assign higher importance to key terms, even if they are sparsely represented in the given paragraph. The text below illustrates an example demonstrating the query generation process using the doc2query/all-t5-base-v1 model. In this example, the article context is fed as input to the model, which generates a set of queries as output.

Input_text: Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural

Generated Queries:

1: what is the difference between artificial intelligence and human intelligence?
2: what is artificial intelligence?
3: what is artificial intelligence?
4: what is Ai MWS definition?
5: what is the definition of artificial intelligence?
6: what is AI a human or a machine?
7: what is AI research?
8: what is AI a machine or a human?
9: why I am doing AI research?
10: what is artificial intelligence in psychology?

5.4 Searching module

The Searching module in our implementation is developed using Java programs and utilizes the Jsoup Java library. Jsoup provides a convenient API for our agents to browse the Internet, offering features for fetching URLs, extracting data, and manipulating HTML using HTML5 DOM methods and CSS selectors. Furthermore, we have an article collection agent dedicated to aggregating the downloaded articles.

Table 2 presents information about 10 academic search engines that we crawled, namely ACMDL1, Base2, CiteSeerX3, MDPI4, Nature5, SciELO6, TaylorFrancais7, ResearchGate8, PubMed Central9, and Eric10. Each agent is assigned to crawl a specific site within the academic search engine and extract article information based on the queries received from the query buffer.

Table 3 displays the data obtained through web crawling, including title, type, authors, journal, publication date, access link, citation information, abstract, keywords, and references. The table also accounts for information we could not gather through crawling from academic search engines.

Table 2. Academic search engine crawled from

	Title	Type	Authors	Journal	Publiched In	Link Access	Citation	Abstract	Keywords	References
ACMDL	√	√	√	√	√	√	√	√	√	√
Base	√	√	√	√	√	√	√	√	√	√
CiteSeerX	√	√	√	√	√	√	√	√	√	√
MDPI	√	√	√	√	√	√	√	√	√	√
Nature	√	√	√	√	√	√	√	√	√	√
SciELO	√	√	√	√	√	√	√	√	√	√
TaylorFrancais	√	√	√	√	√	√	√	√	√	√
Researchgate	√	√	√	√	√	√	√	√	√	√
PubMed Central	√	√	√	√	√	√	√	√	√	√
Eric	√	√	√	√	√	√	√	√	√	√

Table 3. Crawled information from academic search engine

Website Name	Type	Year	Types of Documents Covered	Size
ACMDL	Digital library	July 1997	Journal articles, conference proceedings, Theses, newsletters and books	2.8+ million articles
Base	Search Engine	June 24, 2004	Journal articles, conference proceedings, Theses, newsletters and books	136 million articles
CiteSeerX	Digital library	1997	Journal and transaction articles, technical reports, books	5+ million scholarly documents
MDPI	Digital library	1996	Journals, research articles, reviews, book reviews	386 peer-reviewed journals
Nature	Digital library	1869	Journal articles, magazines, news, books, reviews	800,000+ articles
SciELO	Digital library	1997	Journal articles, original works, case reports, technical reports, reviews	1723 journals
TaylorFrancais	Digital library	June 2011	Journal articles, ebooks	4,762,000+ articles
Researchgate	Digital library	May 2008	Journal articles, conference proceedings, Theses, newsletters and books	135+ million publications
PubMed Central	Digital library	February 2000	Journal articles	5.1 million articles
Eric	Digital library	1966	Journal articles, conference proceedings, Theses, newsletters and books	1.3 million items

To extract essential information from a document, we select specific sections that accurately represent the paper's main contribution. This approach helps filter out irrelevant or tangential parts, ensuring that the chosen sections effectively describe the core content.

5.5 Dataset preparation and learning module

5.5.1 Learning model

For the learning model, we utilized the OneVsRestClassifier for multilabel classification. The OneVsRestClassifier, or one-vs-all, is a strategy where a separate classifier is trained for each class in a multiclass classification problem. Each classifier is trained to distinguish its corresponding class from all the other classes combined. This approach offers computational efficiency as it requires only n classifiers, where n represents the number of classes in the problem. One notable advantage of the OneVsRestClassifier strategy is its interpretability. Since each class has its own dedicated classifier, it becomes possible to gain insights and understanding about a specific class by examining its corresponding classifier. The OneVsRestClassifier strategy is widely recognized as the most commonly used approach for multiclass classification tasks and is a reliable default choice.

We used three multi-label multi-output classification models: SGDClassifier, LogisticRegression, and SVM model, to adopt the best model in terms of performance. After assessing these models, we aimed to select the model that produced the most optimal results.

5.5.2 Input model preparation

For the input of our classification model, we employed the NLTK (Natural Language Toolkit) Python library to preprocess the article context. Initially, we converted the articles' context to lowercase to ensure consistency in the text representation. Then, we removed numbers, punctuation marks, and whitespace from the article context. This helped eliminate non-alphabetic characters that may not contribute to the classification task. Following that, we utilized the default set of stop words provided by the NLTK library to remove commonly occurring words in the English language that typically do not carry significant meaning for our classification task. Stop words such as "the," "is," and "and" were eliminated to reduce noise in the text data. Lastly, we performed stemming, which transforms words with similar semantics into a standard form.

Once the preprocessing step is complete, we encode the context of the preprocessed articles. We utilize the all-MiniLM-L6-v2 transformer model from the sentence-transformers Python library to accomplish this. This allows us to obtain embedding vectors that are prepared and suitable for use as input in the learning classification model. The provided dataset in Table 4 is a sample dataset used in the learning model. This dataset was acquired from the search module and included the article title and abstract as the context. It is important to note that this dataset is in its raw form, before undergoing preparation steps such as preprocessing and embedding. On the other hand, the subsequent table, labeled Table 5, showcases the same dataset after preparation, including preprocessing and embedding. This processed dataset is now suitable to be used as input for the classification learning model.

Output model preparation

To apply the OneVsRestClassifier for multilabel classification tasks, it is essential to format the output labels (in this case, citations) as a 2D binary (0/1) matrix. This process, known as binarization, can be accomplished using techniques like one-hot encoding and the make_column_transformer transformers in Python. These methods enable the conversion of the label matrix into a binary representation suitable for subsequent machine learning operations. Table 6 presents the citations' IDs before binarization, while Table 7 showcases the citations' IDs after the binarization process. Please note that the provided information is a revised version of the original text. It is important to proofread and ensure the accuracy of the content before finalizing it.

One-Hot Encoding (OHE) is a widely used in data mining tasks to convert categorical features into numerical representations. It allows us to transform a single variable with 'n' observations and 'd' distinct values into 'd' binary variables. Each binary variable represents one of the distinct values, and its presence is indicated by 1 while its absence is indicated by 0 [14]. This preprocessing step is crucial as many machine

learning models require numerical data for effective training and prediction. The One-Hot Encoding process helps capture the categorical information in a format that machine learning algorithms can easily understand and process. It expands the original categorical variable into multiple binary variables, where each variable represents a unique category. Doing so eliminates the inherent ordinality in categorical variables and allows the model to treat each category equally. Table 7 provides an illustrative example of how One-Hot Encoding works, showcasing the transformation of categorical variables into binary variables.

Final Prepared Dataset

The combined datasets from the previous tables are consolidated in Table 8. This table serves as the input for a multi-label, multi-output classification learning model. The articles' context undergoes preprocessing and encoding to be utilized as input for the model. On the other hand, the citations are binarized and employed as the model's output.

Table 4. Articles context before preprocessing and embedding

Doc.ID	Raw.Abstract	Raw.Title
1	To elucidate the organizational and…	The metabolic world of Escherichia…
2	Advanced technologies and biology have…	Reverse Engineering of Biological…
3	The study of networks pervades all of…	Exploring complex…
4	Comprehensive protein protein interaction…	Comparative assessment…
5	The small-world phenomenon â the…	Navigation in a small…
…	…	…
16976	Evolution is the fundamental physical…	Life is physics evolution as a…
16977	High-throughput sequencing technologies…	Limitations of next-generation…
16978	Accurate functional annotation of…	Accurate inference of transcription…
16979	ABSTRACT: A recent article in BMC…	Software that goes with the flow in…

Table 5. Articles context after preprocessing and embedding

ID	Context Embedding
0	(-0.057955895, -0.015522554, -0.07978955, 0.0…
1	(-0.053719852, -0.03363603, -0.0074666627, -0…
2	(-0.05078858, -0.116804756, 0.02499838, 0.045…
3	(0.00081650005, -0.06931782, 0.013128192, -0.0…
4	(0.10938609, -0.011689055, 0.03252615, 0.00642…
…	…
16975	(-0.10026872, -0.041010633, -0.009426038, 0.08…
16976	(-0.12650475, -0.04345017, -0.008223459, -0.039…
16977	(-0.005618644, -0.041031037, -0.054177392, -0…
16978	(-0.104945965, -0.079822, -0.08484905, -0.0903…
16979	(-0.019475678, -0.06901016, 0.0040075155, -0.0…

Table 6. Citations-ID before binarization

ID	ID_CIT_1	ID_CIT_2	ID_CIT_3	ID_CIT_4	ID_CIT_5	ID_CIT_6	ID_CIT...
0	3	2	485	3284	None	None	…
1	16	42	43	60	113	116	…
2	85	0	4	5	10	11	…
3	0	None	None	None	None	None	…
4	23	2	28	488	918	1200	…
…	…	…	…	…	…	…	…
16975	16	420	6357	6371	9466	12096	…
16976	8	7991	15184	15944	16304	16451	…
16977	14	418	10406	10558	14709	15093	…
16978	0	None	None	None	None	None	…
16979	0	None	None	None	None	None	…

Table 7. Citations-ID after binarization

0	1	2	3	4	5	6	…	33755	33756	33757	33758	33795
0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	1.0
4	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0
…	…	…	…	…	…	…	…	…	…	…	…	…
16975	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0
16976	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	1.0
16977	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0
16978	1.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	1.0
16979	1.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	1.0

Table 8. All prepared dataset: Input and output of learning model

ID	Context Embedding	0	1	2	3	…	33757	33758	33759
0	(-0.057955883, -0.015522492, 0.0…	0.0	0.0	0.0	0.0	…	0.0	0.0	1.0
1	(-0.053719815, -0.033635996, -0…	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0
2	(-0.050788604, -0.11680469, 0.04…	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0
3	(0.0008165396, -0.069317825, -0.0…	1.0	0.0	0.0	0.0	…	0.0	0.0	1.0
4	(0.10938613, -0.011689071, 0.0064…	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0
…	…	…	…	…	…	…	…	…	…
16975	(-0.10026879, -0.041010622, 0.08…	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0
16976	(-0.12650478, -0.043450087, -0.03…	0.0	0.0	0.0	0.0	…	0.0	0.0	1.0
16977	(-0.0056186593, -0.04103095, -0…	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0
16978	(-0.10494599, -0.07982196, -0.09…	1.0	0.0	0.0	0.0	…	0.0	0.0	1.0
16979	(-0.019475667, -0.06901014, -0.0…	1.0	0.0	0.0	0.0	…	0.0	0.0	1.0

Table 9. The top 20 articles ranking by cosine similarity

-	Article Number	Cosine_Similarity
0	424.0	0.714115
1	530.0	0.598394
2	708.0	0.546376
3	816.0	0.398675
4	851.0	0.248307
5	888.0	0.217173
6	875.0	0.196829
7	613.0	0.190989

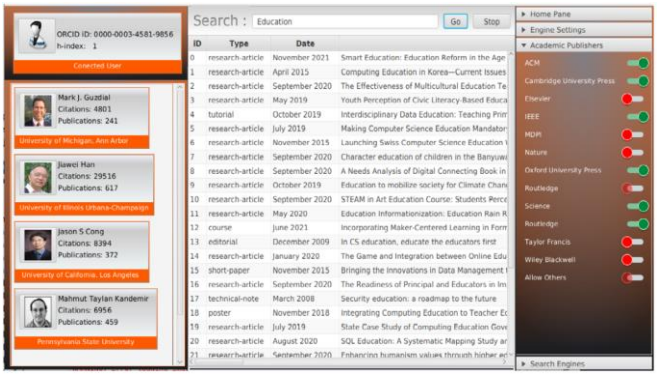

Figure 9. The system GUI

5.6 Recommendation module

In the final stage, we develop the recommendation module using a Python program. This module involves retrieving articles from the article search module's buffer, preparing them for the learning model, and then passing them to the recommendation module. The classification model takes the user's paper as input and predicts a set of associated citations. We apply the multi-output classification model to the user's paper, preparing it in the same way as the input for the learning model (preprocessing and embedding), to predict the citations most relevant to the user's paper. We utilize the cosine similarity function from the sklearn library in Python to filter out the most relevant articles. This function calculates the similarity between each predicted article and the user's paper. We can identify the top N articles with the highest similarity by comparing the similarities. These top N articles are considered the most relevant to the user's paper and are presented as the system's output.

The Scikit-learn library is widely recognized as a highly valuable machine learning library in Python. It offers a range of powerful tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction. Table 9 lists the top 20 recommended articles related to the user's input paper after undergoing the prediction and filtering processes. The articles in the table are ranked based on their cosine similarity scores. This curated list is ready to be returned as the output of the recommendation system.

5.7 The system GUI

We developed the System GUI (Graphical User Interface) using JavaFX, which enables users to interact with our system and access its features (Figure 9). JavaFX provides a comprehensive set of graphical tools that empower developers to design and implement powerful client applications capable of operating seamlessly on multiple platforms.

6. EVALUATION AND RESULTS

We conducted a performance metric evaluation to evaluate the proposed approach's effectiveness. This evaluation technique allowed us to measure the efficacy of the research paper recommendation system.

6.1 Data description

We obtained a comprehensive dataset for our study by collecting articles from various digital libraries using a web crawling system. This dataset contains information such as abstracts, titles, and other details for each article.

The article citations are stored in a file named "citation.dat," while the article contexts are stored in a separate file called "raw-data.csv." Both files follow a specific format for organizing the data:

- In the "citation.dat" file, each line corresponds to the edges connected to a specific node (Figure 10). For example, Line 1 in the file: "2 2295 6231" indicates that two edges are linked to node 0, and their IDs are 2295 and 6231.
- Each line in the "raw-data.csv" file represents a title and abstract associated with a particular node (Figure 11).

This data description provides an overview of the dataset we used in our evaluation, including the organization and content of the files "citation.dat" and "raw-data.csv."

6.2 Data visualisation

Data visualization plays a crucial role in presenting data visually appealing and informatively, making it easier to understand, observe, and analyze. In our evaluation, we utilized Python's powerful libraries for data visualization.

Specifically, we employed the Matplotlib library, a user-friendly and versatile data visualization tool built on NumPy arrays. Matplotlib offers various plot types, including scatter plots, line plots, bar charts, etc. With its extensive functionality, Matplotlib allows us to create visually appealing and customizable visualizations. By utilizing Matplotlib, we could present our evaluation results clearly and visually appealingly, enhancing the understanding and interpretation of the data (Figure 12). Its flexibility and rich feature set make it an ideal choice for data visualization tasks.

6.3 Analysis of the results

Multi-label classification performance metrics: In the case of multi-label classification problems, predictions for each instance consist of multiple labels. The effectiveness of classifiers in such scenarios can be evaluated by calculating the average score of an evaluation metric or by directly comparing the scores for each class. Our research employed two commonly used performance metrics: hamming loss and Jaccard similarity. These metrics provide a robust means to assess the performance of classifiers.

```
2 2295 6231
9 236 876 1799 2115 4534 7011 7081 7089 9972
2 7669 8448
5 1407 1409 2253 3005 5205
3 4112 7339 8276
1 9762
9 1178 4209 4942 5290 6085 6335 8415 9381 12248
2 5044 6335
7 1862 2235 3483 8001 10073 10075 10600
10 247 294 1012 1175 1498 1840 3316 8737 10079 11130
4 6843 7693 10003 14123
7 957 1126 1131 1150 1937 2113 4449
11 546 957 1124 1150 2907 3223 3293 6326 6548 7070 9406
8 2991 4028 5548 5552 7330 8471 9338 11516
11 211 701 1399 1456 1834 2538 2726 3866 4004 4010 11129
```

Figure 10. Citation.dat file

doc.id	Title	Abstract
1	dynamic con	Conditional random fields (CRFs) for sequence modeling have several advanta
2	large n field	We review the holographic correspondence between field theories and string,
3	the largescal	In a cell or microorganism, the processes that generate mass, energy, informat
4	functional ar	The elucidation of the cell's large-scale organization is a primary challenge for
5	xml bioinfor	Motivation: The eXtensible Markup Language (XML) is an emerging standard fc
6	systems biol	To understand biology at the system level, we must examine the structure and
7	computation	Book Description Systems Biology is concerned with the quantitative study of
8	mapping we	Websites of a particular class form increasingly complex networks, and new to
9	understandi	Mobile urban environments present a challenge for context-aware computers
10	shortest patl	Szabó, Alava, and Kertész [Phys. Rev. E 66, 026101 (2002)] consi
11	network bio	A key aim of postgenomic biomedical research is to systematically catalogue al
12	the small wc	Words in human language interact in sentences in non-random ways, and allov
13	metabolomi	Novel techniques for acquiring metabolomics data continue to emerge. Such d
14	genomic ana	Network analysis has been applied widely, providing a unifying language to de
15	fluctuations	Most complex networks serve as conduits for various dynamical processes, ran
16	analysis of w	The connections in many networks are not merely binary entities, either prese
17	inferring net	Naturally occurring networks exhibit quantitative features revealing underlyin
18	metabolomi	In this postgenomic era, there is a specific need to assign function to orphan ge
19	functional ge	The question of whether it is possible to automate the scientific process is of t

Figure 11. Raw-data.csv file

Table 10. Jacard-score for deferents learning model

-	Clf	Jacard Score
0	SGDClassifier	86.1460481100505
1	LogisticRegression	88.24750564339895
2	LinearSVC	88.11109420470825

Table 10 presents the results of applying three classification models (SGD Classifier, Logistic Regression, and SVM) to the dataset. Specifically, it illustrates the Jaccard scores achieved by these models. The classifiers were trained and assessed using a five-fold cross-validation approach to ensure reliable evaluation. This approach involves randomly dividing the dataset into five equal sub-groups. In each iteration, one sub-group serves as the test set, while the remaining four are used for training. The model is then trained on the training set and evaluated on the corresponding test set. This process is repeated until each unique group has been utilized as the test set.

7. SUS EVALUATION

We conducted a usability study using the System Usability Scale (SUS), a well-established questionnaire developed by Brooke [15] to assess various aspects of a system's usability. We introduced the MASSPR tool to multiple groups of researchers and students, allowing them to explore its features and providing them with the SUS questionnaire for feedback. The collected data from the questionnaire revealed that 70% of the participants identified as male, while 30% identified as female.

The SUS questionnaire consists of ten questions that evaluate different dimensions of a system's usability. These dimensions include frequency of use, system complexity, ease of use, need for support, system functions integration, system inconsistencies, learning curve, the cumbersomeness of the system, confidence in the system, and the need for training before use. Upon analyzing the data, we discovered interesting findings from the SUS questionnaire, particularly the positive feedback received from the researchers. As shown in Figure 12, the researchers found the MASSPR tool easy to use and expressed a desire to use it frequently. The feedback also indicated high confidence in utilizing the MASSPR-Tool, with minimal reported inconsistencies. Furthermore, researchers provided valuable suggestions for future versions of the tool, such as expanding its support to include articles from other scientific publishers, as the current version only supports the ACM DL.

The feedback from the usability study highlights the effectiveness of the MASSPR tool and the researchers' overall satisfaction with its usability. These insights will be instrumental in shaping future iterations of the tool, incorporating the requested features, and further enhancing its usefulness in meeting the needs of researchers.

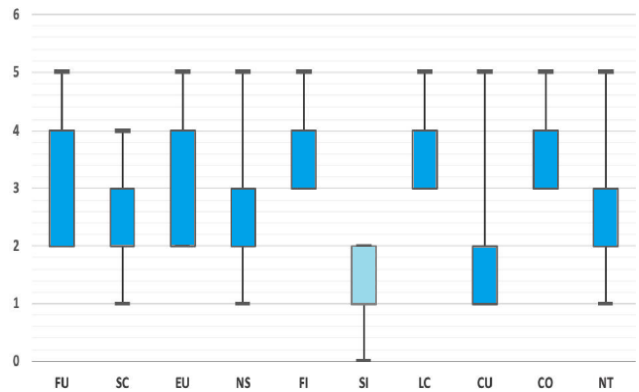

Figure 12. The results of the System Usability Scale questionnaire

8. CONCLUSIONS

This project aims to develop a tool called MASSPR, designed to assist researchers in handling time-consuming daily tasks associated with finding relevant papers for their studies. Instead of manually browsing the web, our tool automates the process by utilizing search engines to retrieve scientific articles. It can search the web, download papers, and apply intelligent mechanisms to filter and recommend the most relevant documents. One key feature of the MASSPR tool is its ability to address the Cold-start issue for new users. By leveraging an innovative approach, our system generates search queries based on a user's given paper or a simple search sentence, assisting new researchers in finding the most relevant documents for their specific needs.

The MASSPR architecture consists of six main modules: the Graphic User Interface (GUI) Front-end modules, GUI Back-end Engine, Query Modelling Module, Searching Module, Filtering Module, and Database Module. The implementation utilizes the Java programming language along with various APIs and libraries. The tool is cross-platform and compatible with Windows, Linux, and macOS. The initial version of our tool has received positive feedback from many users, and incorporating recommendation algorithms has significantly improved the quality of the final recommendations. These findings have substantial implications for researchers seeking to save time by automatically collecting and storing scientific papers on their personal computers. Additionally, the MASSPR tool proves valuable in real-time data set collection of articles.

However, it's important to note that the current version has limitations, such as not fully utilizing the advantages of collaborative-based filtering and disregarding contextual information that influences user preferences as their needs change. For future developments, we plan to expand the tool's capabilities to support the collection of scientific papers from other high-quality digital libraries. Furthermore, we aim to enhance our intelligent recommendation mechanism by addressing the limitations above and exploring the use of deep learning algorithms to improve recommendations further.

REFERENCES

[1] Rachel, M.C. (2018). How scopus powers research solutions for experts worldwide. Elsevier Scopus Blog. https://blog.scopus.com/posts/how-scopus-powers-research-solutions-for-experts-worldwide.

[2] Ujjal Marjit, P.D. (2021). Best 5 academic search engines for research (Multidisciplinary). https://researcherssite.com/best-5-academic-search-engines-for-research-multidisciplinary/.

[3] Deng X.Y., Wang C. (2018). A hybrid collaborative filtering model with context and folksonomy for social recommendation, Ingénierie des Systèmes d'Information (ISI), 23(5): 139-157. https://doi.org/10.3166/ISI.23.5.139-157

[4] Lee, J., Lee, K., Kim, J.G., Kim, S. (2015). Personalized academic paper recommendation system. SRS'15. http://www.joonseok.net/papers/paper_recommend.pdf

[5] Sakib, N., Ahmad, R.B., Haruna, K. (2020). A collaborative approach toward scientific paper recommendation using citation context. IEEE Access, 8: 51246-51255. https://doi.org/10.1109/ACCESS.2020.2980589

[6] Tsolakidis, A., Triperina, E., Sgouropoulou, C., Christidis, N. (2016). Research publication recommendation system based on a hybrid approach. In Proceedings of the 20th Pan-Hellenic Conference on Informatics, New York, USA, pp. 1-6. https://doi.org/10.1145/3003733.3003805

[7] Hanyurwimfura, D., Bo, L., Havyarimana, V., Njagi, D., Kagorora, F. (2015). An effective academic research papers recommendation for non-profiled users. International Journal of Hybrid Information Technology, 8(3): 255-272. https://doi.org/10.14257/ijhit.2015.8.3.23

[8] Bhagavatula, C., Feldman, S., Power, R., Ammar, W. (2018). Content-based citation recommendation. arXiv preprint arXiv:1802.08301. https://doi.org/10.48550/arXiv.1802.08301

[9] Dai, T., Zhu, L., Cai, X., Pan, S., Yuan, S. (2017). Explore semantic topics and author communities for citation recommendation in bipartite bibliographic network. Journal of Ambient Intelligence and Humanized Computing, 9(4): 957–975. https://doi.org/10.1007/s12652-017-0497-1

[10] Ma, X., Wang, R. (2019). Personalized scientific paper recommendation based on heterogeneous graph representation. IEEE Access, 7: 79887-79894. https://doi.org/10.1109/ACCESS.2019.2923293

[11] Meilian, L., Xudan, W., Jie, G., Yan, S. (2015). Ahits-upt: A high quality academic resources recommendation method. In 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, pp. 507-512. https://doi.org/10.1109/SmartCity.2015.120

[12] Shi, H., Ma, W., Zhang, X.L., Jiang, Y.Y. Liu, Y.B., Chu, S.J. (2020). A hybrid paper recommendation method by using heterogeneous graph and metadata. In 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, pp. 1-8. https://doi.org/10.1109/IJCNN48605.2020.9206733

[13] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. https://doi.org/10.48550/arXiv.1810.04805

[14] Ul Haq, I., Gondal, I., Vamplew, P., Brown, S. (2019). Categorical features transformation with compact one-hot encoder for fraud detection in distributed environment. In Data Mining: 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, pp. 69-80. https://doi.org/10.1007/978-981-13-6661-1_6

[15] Brooke, J. (1996). Sus: A quick and dirty'usability. Usability Evaluation in Industry, 189(3): 189-194.