Automating the Collection, Display, Summarization and Podcasting of Academic Research

Tristan Narine and Patrick Hosein
Department of Electrical and Computer Engineering
The University of the West Indies
St. Augustine, Trinidad
tristan.narine@my.uwi.edu, patrick.hosein@uwi.edu

***Abstract*—The research produced by many universities is typically not easily accessible by the average citizen (the Ivory Tower phenomenon). One reason for this is that Faculty members do not regularly update their research publications on the University's web site. Another reason is that, even if the research impacts the lives of citizens, it may not be in a format that can be easily understood by the average citizen. Finally, many people these days prefer a summary of a publication and some may even prefer just to have an audio podcast. We introduce a platform to address these issues and provide a proof of concept using data for a department from a University. The platform performs the following functions for a given department, (1) It periodically scrapes publications for each department member using their Google Scholar page, (2) For each member it provides a list of their publications ranked by date or citations and does the same for the entire department, (3) It uses AI tools to provide a summary of each publication in layman's terms and (4) It produces an audio podcast of the latest publications produced by department members. We are also planning to use a Large Language Model so that citizens can ask questions about the research produced by the department. By automating these processes, the system offers a scalable and low-maintenance solution to increase research visibility, foster public engagement, and improve accessibility to cutting-edge academic advancements.**

***Index Terms*—Web Scraping, Google Scholar publications, Research visibility, AI summarization**

I. Introduction

Academic institutions play a major role in advancing knowledge and addressing problems and challenges through research. However, the impact of research is often poorly communicated [1], both to the public and students who may consider it a career path. It is up to individual researchers to painstakingly place their research information on the departmental websites. This typically requires manual copy and paste from Google Scholar onto the university's website. Consequently, such lists quickly become dated and, because of this, can't act as useful resources for interested students, peers, and the public. The importance of this work lies in its potential to significantly enhance the visibility and accessibility of academic research.

Google Scholar allows universal access to scientific research papers on just about anything and just about every researched publication is uploaded to Google Scholar. Web scraping is a technique used to gather data from multiple websites and compile it into a single spreadsheet or database, simplifying the analysis and visualization of the information [2], [3]. This study proposes an automated approach for collecting, processing, and displaying departmental research publications from Google Scholar, addressing the limitations of manual updates. The solution involves a Python script written to periodically retrieve and process publication data from Google Scholar for all department members, enabling a dynamic and timely display of research done. The system is designed to feature the most recent and most cited works. Additionally, recent publications are summarized using AI to produce layman-friendly explanations. By doing so, this work addresses the challenge of maintaining up-to-date and accurate records of researcher publications, ensuring access to those interested in the latest academic advancements of the department.

This paper presents the design and implementation of the automated publication display system and discusses the technical considerations and decisions made along with the methodology involved. Generally, by automating the gathering and presentation of publications, the proposed approach offers a highly scalable and low-maintenance solution for greatly improving research visibility and access, thereby fostering public and student interest in research activities.

II. Literature Review

A. Web Scraping of Academic Research

Web scraping is a key method of automatic extraction of structured and unstructured web data, and thus a key method of extracting scholarly research from websites like Google Scholar. The traditional manual methods of data acquisition are marked by their time-consuming nature, proneness to human error, as well as general inefficiency, especially with the rising volume of research output. Automated systems overcome these constraints by utilizing target-specific tools and methods adapted to the target website organization. Google Scholar is a significant database of scholarly material and a highly prevalent target for web scraping. It has been proven through studies that automated tools can successfully harvest metadata, citations, and references to scholarly papers [1], [2], [4].

Custom web scrapers are used to harvest specific data sets from websites based on customized criteria. Octobot [2] is a custom web scraper specifically implemented for Google Scholar data harvesting for Applied Science University. They emphasized the advantages of custom solutions, such as the elimination of redundancy and adaptive crawling mechanisms, to meet specific institutional needs. The use of rating systems to prioritize accurate and relevant pages further enhanced the crawler's effectiveness.

Beautiful Soup is widely recognized for its parsing of HTML and XML. It is most suitable for static content scraping, where the layout of the page remains constant independent of requests. Web scraping using Beautiful Soup [4] demonstrated how Beautiful Soup was employed in scraping academic data, including titles, abstracts, and authors. Key advantages ar Ease of Use (Simple syntax makes it simple for new users), Flexible Parsing (Supports tag and attribute-based data extraction) and Integration (The result can be formatted as JSON, CSV, or databases for downstream processing). Weaknesses of Beautiful Soup include inefficiency when parsing JavaScript-dense dynamic pages, where additional tools such as Selenium become necessary [1], [4].

Dynamic content, which is executed with the help of JavaScript, is problematic for static web scraping software. Selenium overcomes this by simulating browser interactions to replicate user actions. Selenium was utilized [5] to scrape Google Scholar through automating navigation, form submission, and log-in. However, Selenium has constraints in terms of operation speed and resource consumption, especially handling large scraping operations [4], [5].

Application Programming Interfaces (APIs) provide a structured and efficient way of acquiring data. In contrast to traditional web scraping, where HTML content interpretation is required, APIs provide data in structured formats like JSON or XML, and applications can easily consume them. Research titled *Scraping Google Scholar Data Using Cloud Computing Techniques* illustrated that by integrating API calls into cloud-based architectures, the retrieval and categorically grouping of scholastic data in real-time was feasible [3]. However, APIs restrict the quantity of data [3], follow a subscription-based model with usage quotas, and are provider-dependent.

B. Web Development Frameworks

Python web frameworks such as Django and Flask are popular for web development because they are easy to use, scalable, and efficient. Django is a "batteries-included" framework, i.e., it comes with in-built tools for database manipulation, authentication, and URL routing, making it suitable for large applications. Large websites such as Instagram, NASA, and Mozilla utilize Django because of its strong architecture and scalability [6], [7].

Flask is a lightweight and minimalist framework that provides developers greater control over libraries and other components. In contrast to Django, Flask falls outside the purview of integrated database systems or form validators, thus providing room for additional customizations. The modular design and inherent simplicity of Flask make it ideally suited for use in microservices and small applications [7].

C. Summarization Techniques

Summarization techniques can be summarized as:

1) **Extractive Summarization:** Chooses prominent sentences from source documents directly based on frequency, position, or BERT embeddings to identify semantic relationships [5], [8].
2) **Abstractive Summarization:** Produces paraphrased summaries based on deep linguistic comprehension. Transformer models such as Pegasus and BART perform well in long-text summarization because of their attention mechanisms [8], [9].
3) **Hybrid Approaches:** Blend extractive precision and abstractive fluency. For example, SciBERT with LED-Large extracts information and produces coherent summaries [9].
4) **Transformer Models:** Models such as T5 and BERT improve summarization by contextualizing text via self-attention. T5's text-to-text model can generate short, useful summaries [5], [8].
5) **NLP Techniques:** Techniques such as NER, TF-IDF, and LDA aid summarization by identifying prominent subjects and entities, which complement transformer models [5], [8].

III. Method

This section outlines the methodology used to automate the collection, processing, and display of publication data from Google Scholar for a university department website. The approach involves three primary stages: data collection, data processing, and data display, each leveraging Python and the Beautiful Soup library for web scraping.

A. Data Collection

Google Scholar does not provide an official API, however, there are third-party APIs available online that can be used. While these APIs (such as SerpAPI, Apify, ScaleSERP, and others) are superior in terms of speed and functionality, they are quite costly and rate-limited. The Beautiful Soup Python library was instead chosen for this project due to its, customizable data extraction, adaptability to website changes, and effectiveness in parsing HTML and XML, allowing for structured extraction of publication data. In the code snippet below, Beautiful Soup is employed to parse the HTML content of a conference paper webpage and extract specific details about the papers listed. By utilizing the find all method, it identifies rows representing individual papers, and with find, it extracts information such as the title, authors, publication date, citation count, and publications from the relevant HTML tags. This data is then organized and stored in a structured format for further use.

A class was created in Python to store each attribute unique to each researcher. Researcher names and unique Google Scholar IDs are stored and read from a text file (format: "first and last name, Google Scholar ID", example: "John Doe,IJJe535"), ensuring flexibility and scalability. This allows the system to be efficiently updated in the event of an addition or removal of researchers without modifying the script.

Beautiful Soup was used to iterate through each researcher's profile, collecting the publication data and saving it in a structured format for processing later. This project requires the collection of each researcher's most recent and most cited publications. This was accomplished using different links within the Beautiful Soup script. The data for the most recent and most cited publications were stored in separate JSON files. Storing scraped data in a JSON file because it's simple, human-readable, lightweight, and portable. JSON's key-value structure naturally fits hierarchical scraped data, and it's widely supported by programming languages. Figure 1 provides a flow chart of the scraping process.

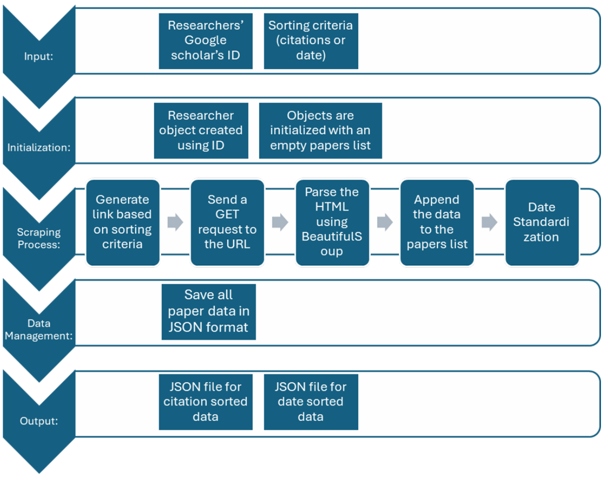

Input:
Researchers' Google scholar's ID
Sorting criteria (citations or date)
Initialization:
Researcher object created using ID
Objects are initialized with an empty papers list
Scraping Process:
Generate link based on sorting criteria
Send a GET request to the URL
Parse the HTML using BeautifulSoup
Append the data to the papers list
Date Standardization
Data Management:
Save all paper data in JSON format
Output:
JSON file for citation sorted data
JSON file for date sorted data

Fig. 1. Overview of scraping process

B. Data Processing

Once the publication data was collected, it underwent a brief cleaning process to ensure uniformity, particularly in date formats. A number of the publication data gathered consisted of different formatted dates such as YYYY/MM/DD, YYYY/MM or just YYYY. To sort these papers in order of most recent, the date parameter was standardized. Papers with missing day and month parameters were assumed to be published on January 1st, and papers with missing day parameters were assumed to be published on the first of the month provided. This was solely used to generate a list of the most recent publications from the department.

The list of most recent publications across the department was generated by initializing an empty list for storing unique papers and a function to track processed titles, ensuring no duplicates. The function iterates through a collection of researchers and their papers, adding papers with unique titles to the list. The papers are then filtered and sorted in descending order of their publication date. Finally, the most recent papers are extracted and stored separately onto another JSON file. This approach ensures the collection of only the latest, unique publications in an organized manner.

For each recent publication, the title and description were merged into a single input, which was then processed using a transformer-based summarization pipeline. The summarization pipeline was initialized using the Hugging Face Transformers library with the "facebook/bart-large-cnn" model. This pre-trained model generated concise summaries of input text by extracting key information, subsequently integrated into the original dataset, ensuring the information remained easily accessible. Additionally, the recent publications served as the foundation for creating a conversational-style podcast using NotebookLM, an AI-powered tool designed to summarize and discuss content, transforming the information into an engaging audio format. This approach makes complex academic research more accessible and understandable, not only for researchers but also for laypeople and other non-experts, fostering broader engagement and knowledge sharing.

C. Email Notification System

The email notification system was developed to boost user interaction and provide timely alerts on scholarly publications. The feature permitted users to subscribe to particular researchers or to notifications on the discovery of new research papers in the database. Subscribers were notified via email upon the discovery of a new publication, thus alerting them to the current advancement in academia. The email alert system was built with the `smtplib` library implemented in Python. A function was called whenever a new paper was found to alert all the users who were subscribed. The Simple Mail Transfer Protocol was implemented while sending the message through a Gmail account created specifically for this function. The system managed subscriptions by allowing users to select researchers that they wished to follow or subscribe to generic new publication announcements. The subscriptions were stored in a JSON file database, linking users to researcher IDs. At scheduled intervals, the system scraped publication details, and on noticing a new publication by a subscribed researcher, the system would send an automated message.

D. Data Display

Several frameworks were considered for the application of this project including Django and Flask. Flask was chosen as it is a lightweight, minimalistic framework that offers flexibility without the overhead of Django's built-in features. Flask's modularity and support for asynchronous handling make it a strong choice for simple, high-concurrency applications, while Django suits larger projects requiring a full-featured, structured framework. A flask-based web application was set up to serve as the display platform for the publication data. The collected data, saved in JSON format is read periodically by the Flask application and dynamically loaded onto the department's website. Key elements of the display include:

1) Home page: Displays the predefined list of all researchers in the department.
2) Individual researcher pages: Showcases the most recent and most cited publications from each researcher.
3) Most recent publications page: Showcases the most recent publications from the department.
4) AI summary page: Showcases the generated AI summary for the most recent publications.

The project was deployed on the server located at `http://dece.ai.tt`. Deployment involved securely transferring project files to the server using an SSH key. The Flask application was then configured to run on the server, enabling online accessibility for testing and further development. The deployment utilized NGINX as a reverse proxy and Gunicorn as the WSGI HTTP server to ensure efficient handling of web traffic and application requests, providing a robust and scalable setup for the web application. Additionally, the Python script responsible for collecting and processing the data is scheduled to run every two days via a cron job, ensuring the dataset remains consistently up to date.

E. System Overview

This system automates the collection, processing, and presentation of academic research data in a continuous cycle:

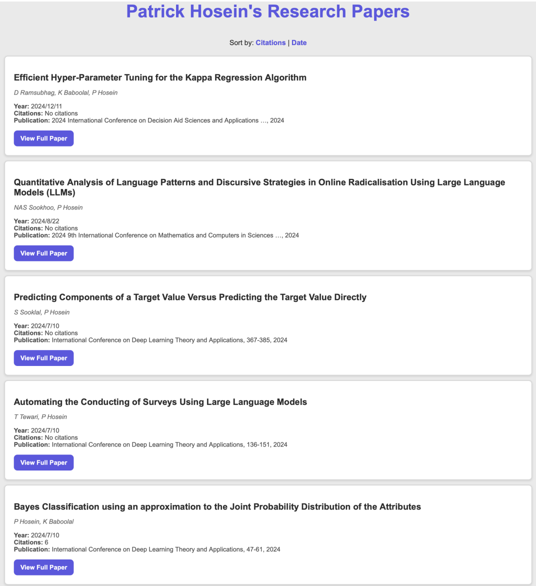
1) **Data Collection**: Google Scholar pages are scraped to gather raw publication data.
2) **Error Handling**: Missing or incomplete data is validated and corrected (Date elements, citation counts, etc).
3) **Data Storage**: Processed data is saved into two JSON files for researchers and departments (sorted by dates or citations).
4) **Publication Filtering**: Most recent publications from the department are identified.
5) **AI Summaries and Podcasts**: Abstracts and titles are summarized using AI, and podcasts are generated for alternative access.
6) **Web Rendering**: Flask dynamically renders data, summaries, and audio on a web application.
7) **48-Hour Cycle**: The system refreshes every 48 hours to ensure updated content.
8) **Email Notifications**: The system sends subscribed used timely notifications depending on subscribed criteria.

This efficient pipeline delivers accurate, up-to-date research data with minimal manual intervention.

IV. Results

A. Data Collection Accuracy

The system accurately scrapes publication data from Google Scholar using Beautiful Soup. The script retrieved publication data for each researcher listed in the department's text file, including titles, authors, publication dates, citation counts, abstracts, and publications. The accuracy of the collected data was validated by comparing a sample of scraped entries with the original Google Scholar profiles. Across 11 researchers, the average data extraction accuracy was 100% and required approximately 90 seconds, demonstrating the efficiency of the approach despite the absence of third-party APIs. A sample of the top 5 papers for a sample researcher from Google Scholar is shown in Figure 2 and the corresponding scraped data as displayed on the platform is provided in Figure 3.



Patrick Hosein's Research Papers

Efficient Hyper-Parameter Tuning for the Kappa Regression Algorithm

D Ramsubhag, K Baboolal, P Hosein

Year: 2024/12/11
Citations: No citations
Publication: 2024 International Conference on Decision Aid Sciences and Applications …, 2024

Quantitative Analysis of Language Patterns and Discursive Strategies in Online Radicalisation Using Large Language Models (LLMs)

NAS Sookhoo, P Hosein

Year: 2024/8/22
Citations: No citations
Publication: 2024 9th International Conference on Mathematics and Computers in Sciences …, 2024

Predicting Components of a Target Value Versus Predicting the Target Value Directly

S Sooklal, P Hosein

Year: 2024/7/10
Citations: No citations
Publication: International Conference on Deep Learning Theory and Applications, 367-385, 2024

Automating the Conducting of Surveys Using Large Language Models

T Tewari, P Hosein

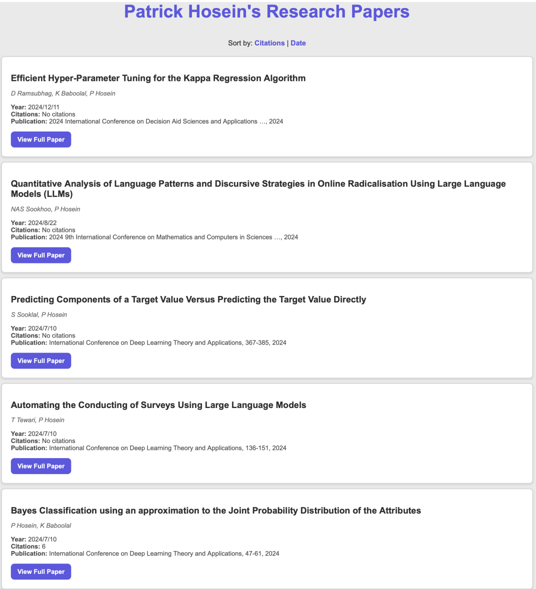
Year: 2024/7/10
Citations: No citations
Publication: International Conference on Deep Learning Theory and Applications, 136-151, 2024

Bayes Classification using an approximation to the Joint Probability Distribution of the Attributes

P Hosein, K Baboolal

Year: 2024/7/10
Citations: 6
Publication: International Conference on Deep Learning Theory and Applications, 47-61, 2024

Fig. 2. Example of Most Recent papers for Researcher from Google Scholar



Patrick Hosein's Research Papers

Efficient Hyper-Parameter Tuning for the Kappa Regression Algorithm

D Ramsubhag, K Baboolal, P Hosein

Year: 2024/12/11
Citations: No citations
Publication: 2024 International Conference on Decision Aid Sciences and Applications …, 2024

Quantitative Analysis of Language Patterns and Discursive Strategies in Online Radicalisation Using Large Language Models (LLMs)

NAS Sookhoo, P Hosein

Year: 2024/8/22
Citations: No citations
Publication: 2024 9th International Conference on Mathematics and Computers in Sciences …, 2024

Predicting Components of a Target Value Versus Predicting the Target Value Directly

S Sooklal, P Hosein

Year: 2024/7/10
Citations: No citations
Publication: International Conference on Deep Learning Theory and Applications, 367-385, 2024

Automating the Conducting of Surveys Using Large Language Models

T Tewari, P Hosein

Year: 2024/7/10
Citations: No citations
Publication: International Conference on Deep Learning Theory and Applications, 136-151, 2024

Bayes Classification using an approximation to the Joint Probability Distribution of the Attributes

P Hosein, K Baboolal

Year: 2024/7/10
Citations: 6
Publication: International Conference on Deep Learning Theory and Applications, 47-61, 2024

Fig. 3. Recent papers from Researcher as displayed on the Platform

B. Summarization and AI Integration

Using the Hugging Face facebook/bart-large-cnn model, abstracts and titles of the most recent publications were summarized into concise descriptions. The performance of the transformer-based model in abstract summarization was measured with the assistance of cosine similarity and ROUGE scores. The average cosine similarity score of 0.659 was achieved, which shows a moderate to high semantic similarity between the true abstracts and the generated summaries. This specific measure is crucial to determine the retention of the overall meaning and major concepts of the abstracts in the generated summaries.

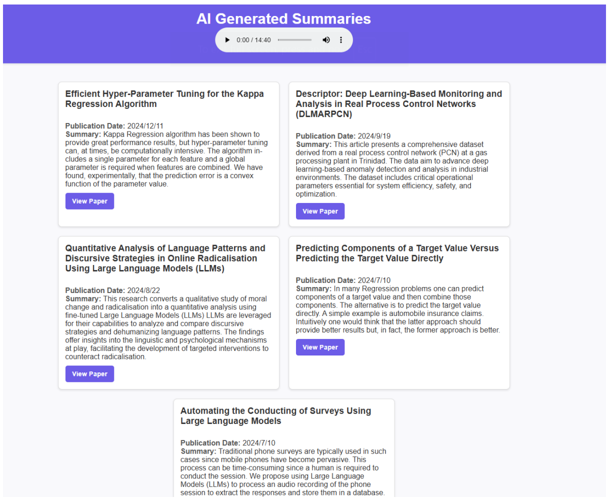
The ROUGE scores (Table I), which are a lexical overlap measure, also reflect the summarization performance of the model. ROUGE-1 (unigram) had a precision of 0.996, recall of 0.408, and F-measure of 0.578. ROUGE-2 (bigram) achieved a precision of 0.962, recall of 0.391, and F-measure of 0.555, while ROUGE-L (longest common subsequence) achieved a precision of 0.991, recall of 0.406, and F-measure of 0.575. The high precision values across all ROUGE metrics indicate that the summaries were able to maintain significant terms and phrases of the original abstracts. The moderate recall scores, however, indicate that the summaries are concise, and may omit less critical information.

TABLE I
Evaluation Metrics for Abstract Summarization

Metric	Precision	Recall	F-Measure
ROUGE-1 (Unigrams)	0.996	0.408	0.578
ROUGE-2 (Bigrams)	0.962	0.391	0.555
ROUGE-L (LCS)	0.991	0.406	0.575
Cosine Similarity	0.659 (Average)		

C. Web Application

The Flask-based web application displayed publication data dynamically on the department's website. Key features included: A homepage listing all researchers. Individual pages displaying the most recent and most cited publications, most recent publications page and AI summary integration for ease of understanding.



AI Generated Summaries

Efficient Hyper-Parameter Tuning for the Kappa Regression Algorithm

Publication Date: 2024/12/11
Summary: Kappa Regression algorithm has been shown to provide great performance results, but hyper-parameter tuning can, at times, be computationally intensive. The algorithm includes a single parameter for each feature and a global parameter is required when features are combined. We have found, experimentally, that the prediction error is a convex function of the parameter value.

Descriptor: Deep Learning-Based Monitoring and Analysis in Real Process Control Networks (DLMARPCN)

Publication Date: 2024/9/19
Summary: This article presents a comprehensive dataset derived from a real process control network (PCN) at a gas processing plant in Trinidad. The data aim to advance deep learning-based anomaly detection and analysis in industrial environments. The dataset includes critical operational parameters essential for system efficiency, safety, and optimization.

Quantitative Analysis of Language Patterns and Discursive Strategies in Online Radicalisation Using Large Language Models (LLMs)

Publication Date: 2024/8/22
Summary: This research converts a qualitative study of moral change and radicalisation into a quantitative analysis using fine-tuned Large Language Models (LLMs) LLMs are leveraged for their capabilities to analyze and compare discursive strategies and dehumanizing language patterns. The findings offer insights into the linguistic and psychological mechanisms at play, facilitating the development of targeted interventions to counteract radicalisation.

Predicting Components of a Target Value Versus Predicting the Target Value Directly

Publication Date: 2024/7/10
Summary: In many Regression problems one can predict components of a target value and then combine those components. The alternative is to predict the target value directly. A simple example is automobile insurance claims. Intuitively one would think that the latter approach should provide better results but, in fact, the former approach is better.

Automating the Conducting of Surveys Using Large Language Models

Publication Date: 2024/7/10
Summary: Traditional phone surveys are typically used in such cases since mobile phones have become pervasive. This process can be time-consuming since a human is required to conduct the session. We propose using Large Language Models (LLMs) to process an audio recording of the phone session to extract the responses and store them in a database.

Fig. 4. Sample AI Summaries of Publications

D. Email Notification System

The email notification system was successfully integrated and tested within the research update cycle. Users are able to subscribe to multiple researchers, and notifications are sent automatically upon the detection of new publications. The automated nature of the system reduced the need for manual intervention, ensuring efficient update dissemination.

V. Discussion

The automatic system for gathering, summarizing, and presenting academic research has shown its potential in enhancing research visibility and accessibility. Through the combination of web scraping, AI-powered summarization, and a dynamic web interface, the system provides a viable solution to the problem of keeping publication records up to date. Although the system has produced encouraging results, some limitations and areas of improvement need to be recognized. A significant drawback of web scraping is that it relies on the web page structure of Google Scholar. Any change in the composition of Google Scholar or prohibition of automatic access would adversely affect the system's effectiveness in obtaining publication data. Future versions of the system can overcome this vulnerability through the implementation of adaptive scraping methods or by employing alternative data procurement approaches, such as the potential integration of third-party APIs where applicable. The summarization task, dominated by transformer-based models, has been successful in summarizing research abstracts into more readable summaries. The evaluation metrics show that although the precision of the summaries is high, the recall is moderate, which means that some secondary information may be omitted. Although this is consistent with the objective of brevity, improvements are possible to achieve a more desirable trade-off between brevity and completeness. Fine-tuning the summarization model or using a hybrid approach that combines both extractive and abstractive techniques can additionally enhance the coherence of the generated summaries. The web application, developed utilizing the Flask framework, offers a user-friendly and accessible environment for individuals to investigate research publications. Its inherently dynamic characteristics guarantee that the information presented is consistently up-to-date, thereby alleviating the administrative workload associated with manual updates. Furthermore, supplementary features, including enhanced search functionalities, interactive data visualizations, and mechanisms for user feedback, have the potential to significantly enhance user engagement and overall functionality. The incorporation of email alerts significantly enhanced research visibility by the immediate communication of updates to the respective users. This feature obviated the need for manual checks to access new research, thereby facilitating interaction and usability within the system. Not only does automation of this process improve efficiency, it also encourages the creation of an active academic community, as pertinent research can be disseminated to the respective audience in real time.

A. Limitations

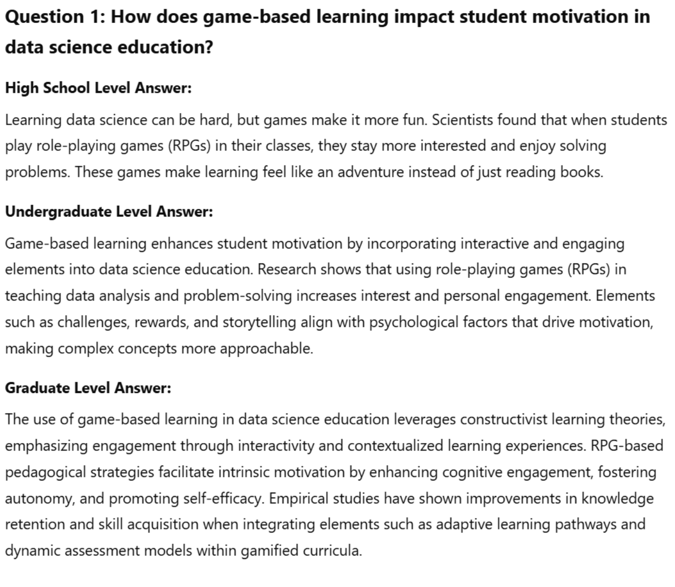
To address the system's limitations, improving robustness through automated error-handling and real-time monitoring of scraping logic is essential. Enhancing the summarization pipeline via model fine-tuning or reinforcement learning would provide improved precision-recall tradeoff, producing more precise summaries. The system can further be made to analyze citation networks, collaboration graphs, and research impact, and provide actionable insights for institutional decision-making. Adding interactive elements like advanced search capabilities, user feedback forms, and data visualizations would enhance user engagement and functionality as a whole.

B. Broader Implications

This system provides a scalable method for automating academic data gathering and presentation that can be used across different departments of any university. With inclusion of functions such as AI-created summaries and dynamic web interfaces, it is possible for it to be used as an integrated tool for monitoring and displaying research output irrespective of the field. For STEM departments, it can monitor high-impact publications and work together. Humanities and social science departments can utilize this system to open themselves up more and encourage engagement from broader audiences. This flexibility makes the system's usability across a wide range of academic disciplines, fostering departmental and university accessibility, collaboration, and visibility.

C. Future Work: Automated Query System

An AI-based query system can help users navigate subjects and get brief summaries of related studies, tailored to their academic level. Transformer-based models are effective at summarizing scientific articles [10], and AI-based tools can optimize literature reviews by finding and combining pertinent studies [11]. Generative AI is also promising for improving research retrieval systems, although issues such as hallucinations need to be overcome [12]. Both the public and the university would find such a feature helpful. Students and individual researchers would be able to view summaries that are appropriate for their level of experience, which would improve learning and engagement with academic material. Universities can use it to facilitate research work by helping staff and students to scan relevant literature quickly. Future work could integrate real-time search, fine-tune AI models for personalized academic summarization, and optimize search algorithms, making literature reviews more accessible.



Question 1: How does game-based learning impact student motivation in data science education?

High School Level Answer:

Learning data science can be hard, but games make it more fun. Scientists found that when students play role-playing games (RPGs) in their classes, they stay more interested and enjoy solving problems. These games make learning feel like an adventure instead of just reading books.

Undergraduate Level Answer:

Game-based learning enhances student motivation by incorporating interactive and engaging elements into data science education. Research shows that using role-playing games (RPGs) in teaching data analysis and problem-solving increases interest and personal engagement. Elements such as challenges, rewards, and storytelling align with psychological factors that drive motivation, making complex concepts more approachable.

Graduate Level Answer:

The use of game-based learning in data science education leverages constructivist learning theories, emphasizing engagement through interactivity and contextualized learning experiences. RPG-based pedagogical strategies facilitate intrinsic motivation by enhancing cognitive engagement, fostering autonomy, and promoting self-efficacy. Empirical studies have shown improvements in knowledge retention and skill acquisition when integrating elements such as adaptive learning pathways and dynamic assessment models within gamified curricula.

Fig. 5. Example of Research Query using OpenAI's ChatGPT

VI. Conclusion

This paper outlines a robust methodology for automating the collection, summarization, and display of academic research data, addressing challenges associated with manual updates and limited accessibility. By integrating web scraping, AI summarization, and a Flask-based web application, the system successfully enhances research visibility and accessibility while reducing administrative effort. Despite limitations such as reliance on web scraping and moderate summarization recall, the system demonstrates its potential to transform how academic institutions manage and disseminate research outputs. Future improvements, including the integration of APIs, enhanced error-handling, and model fine-tuning, will further improve the system's accuracy and scalability, ensuring its adaptability to evolving technological and institutional needs.

References

[1] D. PRATIBA, A. M.S., A. DUA, G. K. SHANBHAG, N. BHANDARI, and U. SINGH, "Web scraping and data acquisition using google scholar," in *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, pp. 277–281, 2018.

[2] M. A. Aljemazi and M. A. Khder, "Octobot - web scarping towards retrieving google scholar data," in *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, pp. 477–482, 2022.

[3] N. A. Sultan and D. B. Abdullah, "Scraping google scholar data using cloud computing techniques," in *2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, pp. 14–19, 2022.

[4] S. Pant, E. N. Yadav, Milan, M. Sharma, Y. Bedi, and A. Raturi, "Web scraping using beautiful soup," in *2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS)*, vol. 1, pp. 1–6, 2024.

[5] I. Naing, N. Funabiki, K. H. Wai, and S. Thandar Aung, "A design of automatic reference paper collection system using selenium and bert model," in *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*, pp. 267–268, 2023.

[6] M. Sharma, M. S. Khan, and J. Singh, "Python & django the fastest growing web development technology," in *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, pp. 1–9, 2024.

[7] P. Thakur and P. Jadon, "Django: Developing web using python," in *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 303–306, 2023.

[8] N. Radha, R. Swathika, K. R. Uthayan, and M. K. B, "Ai-driven summarization of academic literature using transformer model," in *2024 Second International Conference on Inventive Computing and Informatics (ICICI)*, pp. 359–364, 2024.

[9] Z. Dar, M. Raheel, U. Bokhari, A. Jamil, E. M. Alazzawi, and A. A. Hameed, "Advanced generative ai methods for academic text summarization," in *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, pp. 1–7, 2024.

[10] R. Haruna, A. Obiniyi, M. Abdulkarim, and A. Afolorunsho, "Automatic summarization of scientific documents using transformer architectures: A review," in *2022 5th Information Technology for Education and Development (ITED)*, pp. 1–6, 2022.

[11] P. Castillo-Segura, C. Alario-Hoyos, C. D. Kloos, and C. Fernández Panadero, "Leveraging the potential of generative ai to accelerate systematic literature reviews: An example in the area of educational technology," in *2023 World Engineering Education Forum - Global Engineering Deans Council (WEEF-GEDC)*, pp. 1–8, 2023.

[12] M. Saied, N. Mokhtar, A. Badr, M. Adel, P. Boles, and G. Khoriba, "Ai in literature reviews: a survey of current and emerging methods," in *2024 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 61–65, 2024.