Recommendation Module Based on Web Scraping and LLM Models with the RAG Technique

1st Mohamed Hachicha
ESPRIT School of Engineering, Tunisia
hachicha.mohamed@esprit.tn

2nd Mohamed Aziz Omezine
ESPRIT School of Engineering, Tunisia
mohamedaziz.omezine@esprit.tn

3rd Mohamed Khalil Zghal
ESPRIT School of Engineering, Tunisia
mohamedkhalil.zghal@esprit.tn

4th Mohamed Hedi Riahi
ESPRIT School of Engineering, Tunisia
mohamedhedi.riahi@esprit.tn

5th Lotfi Ncib
RidchaData, France
lotfi.ncib@ridchadata.com

***Abstract*—In this research, we investigate the implementation of Retrieval-Augmented Generation (RAG) technology to recommend suitable engineering internships to individuals who are looking for an internship based on their CVs by improving the search for correspondence between candidates and employers, ensuring that both parties find the best adequacy. This innovative approach leverages web scraping, data preprocessing and advanced machine learning techniques including vector databases designed to handle high-dimentional vector data allowing for searches based on the proximity or similarity of vectors, and embedding methods used to convert high-dimentional data into vector representations that can be stored in these vector databases. By analyzing and understanding the content of each CV and internship offer, our methodology aims to deliver personalized and relevant internship opportunities, enhancing candidate-employer matching.**

***Index Terms*—Retrieval-Augmented Generation (RAG), recommend internships, LLM, machine learning techniques,CVs, web scraping**

I. Introduction

Engineering students face several challenges when seeking work placements at the end of their studies. These challenges include a lack of an established professional network to find opportunities, strong competition for internships, especially at well-known companies, and a mismatch between the skills acquired during their studies and those sought by employers. Furthermore, in many countries, the discovery of internship opportunities still relies on the networks and personal relationships of aspiring interns. To bridge this gap, these individuals strive to build a network through events, online platforms, and informational interviews. Another challenge arises from the fact that students may miss out on opportunities due to their concerns, such as exams or ongoing projects. Additionally, companies may seek skills in emerging tools or technologies that students have not yet learned.

Our goal is to guide students in tailoring their applications to meet specific internship requirements. We focus on the problem of matching engineering students' CVs with placement offers. This is a crucial challenge in education and recruitment, as it involves finding the best possible match between the skills, experience, and aspirations of engineering students and the requirements, contexts, and opportunities offered by companies for end-of-study placements.

The main challenges for students are to find an internship that aligns with their skills and career objectives, given the wide variety of available opportunities and to stay up-to-date with both new and old internship offers. For companies, the major challenge is to identify candidates best suited to their needs and culture. Traditional approaches to solving these problems include participating in recruitment fairs and company forums and using platforms for manual contact between students and companies.

In the realm of artificial intelligence (AI), particularly in recommendation systems, the integration of Collaborative Filtering, Content-Based Models and Deep Learning has significantly enhanced the accuracy and relevance of recommendations. Collaborative Filtering leverages the preferences and behaviors of similar users to suggest items, making it highly effective for personalized recommendations. On the other hand, Content-Based Models analyse the attributes of items and users, such as skills and experiences in CVs, to find the best matches. Deep Learning further amplifies these capabilities by extracting complex feature from vast datasets, enabling a deeper understanding of user preferences and item characteristics. Together, these techniques create robust recommendation systems that can adapt to individual needs, providing highly tailored and precise suggestions in various domains, from job matching to personalized content delivery.

However, we propose modern approaches based on AI which allows internship seekers to both find a suitable internship based on their skills and stay informed about internship opportunities that might otherwise be missed. Additionally, it helps improve their CV so that it aligns better with the top internship offers by providing guidance on what to avoid and what to add. These approaches aim to first perform a semantic analysis of CVs and internship offers, then develop recommendation algorithms based on machine learning, and finally use Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) to extract and interpret relevant information [1]–[3], [11].

LLMs provide a contextual understanding of CVs and

2024 Artificial Intelligence Revolutions (AIR) | 979-8-3503-7788-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/AIR63653.2024.00025

979-8-3503-7788-0/24/$31.00 ©2024 IEEE
DOI 10.1109/AIR63653.2024.00025

internship offers. They have an advanced ability to grasp the context and nuances of natural language, enabling precise semantic analysis of CVs and internship descriptions, capturing skills and experience even if they are not explicitly mentioned in the same way.

A crucial step is extracting information from CVs and offers. LLMs can extract and interpret relevant information, identifying candidates' skills, experience, and aspirations, as well as companies' requirements and expectations. Thanks to their deep understanding of language, LLMs can tailor recommendations more finely, considering the nuances and subtleties of candidate profiles and placement descriptions [1], [4], [6], [10].

In our work, we will combine LLMs with RAG for matching CVs and internship offers. While LLMs excel at understanding and generating natural language, they can lack precision in integrating structured data. RAG addresses this by combining text generation with the retrieval of precise, structured information from databases or documents.

RAG can retrieve specific, relevant information from a database or corpus of documents, enhancing the accuracy of the responses generated by LLMs. This is particularly useful for matching, where it is crucial to find precise matches between candidates' skills and the requirements of placement offers. Additionally, RAG can access recent and up-to-date information, ensuring that recommendations and matches are based on the most current data available. This is important in a dynamic environment where company requirements and candidate skills change rapidly.

Using RAG, LLMs can query large and varied databases, enabling the efficient and accurate processing of large volumes of CVs and placement offers. The integration of RAG improves the accuracy of matches by verifying and validating extracted information, reducing the risk of inaccurate or irrelevant recommendations [1]–[3], [9].

The combined use of LLMs and RAG for matching engineering students' CVs with internship offers significant advantages. LLMs provide advanced contextual understanding and text generation capability, while RAG enhances accuracy and efficiency by retrieving specific, up-to-date information. This combination creates a more robust, accurate, and adaptable recommendation system, optimally meeting the needs of both students and companies.

Effective matching between CVs and internship offers is crucial, as it can significantly impact the early careers of engineers and companies' ability to innovate and grow. The judicious use of AI and big data could greatly improve this process, benefiting all stakeholders.

II. Related Work

The emergence of large language models (LLMs) has revolutionized human-like language understanding and generation. These models, powered by transformers and other advanced neural architectures, excel in various natural language processing (NLP) tasks like text generation, translation, sentiment analysis, summarization, question answering, coding assistance, and conversational agents. However, their generic nature often leads to standardized responses, hindering personalization – a crucial feature in today's world of high user expectations and limited time [8]–[11]. Personalization empowers LLMs to respond more precisely to individual preferences. By tailoring responses to specific needs, LLMs can deliver significant benefits across various sectors, from improving user experiences to enhancing efficiency in businesses and educational institutions. Choosing the right customization method is vital, as each technique presents unique potential and limitations. Factors like task requirements, resource needs, implementation complexity, and generated response quality all influence this decision. This work focuses on LLM customization techniques, specifically Retrieval-Augmented Generation (RAG). RAG leverages retrieved information to enhance LLM response generation. We apply RAG to address the challenge of matching engineering students' CVs with final project internship offers. This approach provides personalized and relevant internship recommendations by conducting an in-depth analysis of CV and internship content, ultimately improving the matching process [3]–[5], [11].

III. Methodology

We begin by presenting and detailing the key stages of our work during the development of the initial version of our system for matching CVs with placement offers. This process comprises several essential phases:

1) Creating a Web Scraping Script for Job Portals:
   To gather data on available internships, we developed a web scraping script to extract information from several job portals, including Indeed, LinkedIn, ZipRecruiter, and Glassdoor. The script collects data such as job titles, descriptions, requirements, and company details. This automated approach ensures a comprehensive and up-to-date dataset of internship opportunities.
2) Data Preprocessing:
   The raw data collected through web scraping undergoes preprocessing to ensure consistency and quality. This step involves cleaning the data, handling missing values, normalizing text, and performing other necessary transformations to prepare it for further analysis. By standardizing the data, we ensure that the information is reliable and ready for the next stages of the system development.
   We can see in the figure **Fig. 1** the null percentage for each feature and we can conclude that the most critical features for embedding with a lower null percentage include location, title, job_url, is_remote and description.

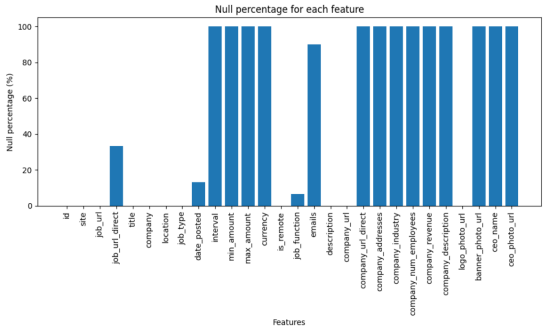

Fig. 1. Null percentage for each features

From the graph in **Fig. 2** we can conclude that the average words Count is 565 words in each description This information can guide the selection of chunk size and chunk overlap when creating text embeddings to ensure that the chunks are manageable and capture meaningful segments of the descriptions.

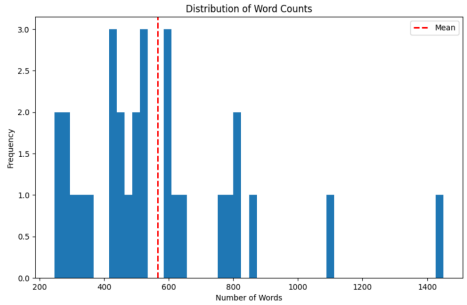

Fig. 2. Distribution of word counts

3) Choosing the Vector Database:
We evaluated different vector databases for storing and managing embeddings. The options considered included Postgres with the pgvector extension, ChromaDB, and Pinecone. After a careful comparison, we selected ChromaDB due to its performance and ease of integration. Our comparison and choice were based on the following parameters [3], [5]:
   - Open source: The software's source code is freely available to the public, allowing developers to review, modify, and distribute it.
   - Self-host: The database can be hosted on a user's infrastructure rather than being dependent on a third-party cloud service.
   - Cloud management: Offers an interface for database cloud management.
   - Purpose-built for vectors: Specifically designed with vector storage and retrieval in mind.
   - Queries per second: Benchmarking how many queries the database can handle per second using the nytimes-256-angular dataset.
   - Latency: The delay in milliseconds between initiating a request and receiving a response, with 95% of query latencies falling under the specified time for the nytimes-256-angular dataset.
   - Supported index types: Various indexing techniques supported by the database, influencing search speed and accuracy.
   - Hybrid search: Allows for combining traditional scalar queries with vector queries.
   - Disk index support: Supports storing indexes on disk, essential for handling large datasets.
   - Role-based access control: Security mechanisms allowing permissions to be granted to specific roles or users.
   - Dynamic segment placement vs. static data sharding: How the database manages data distribution and scaling.
   - Free hosted tier: Availability of a free cloud-hosted version.
   - Pricing: Cost associated with storing and querying specific amounts of data.
4) Comparing Different Embedding Methods:
To find the most suitable embedding method for our application, we compared several options, including different state-of-the-art models. This comparison focused on factors such as accuracy, computational efficiency, and compatibility with our dataset and system requirements.

TABLE I
EMBEDDING METHODS

Model	Pages Per Dollar	Performance	Max input
Text-embedding-3-smal	62500	62.3%	***8191***
Text-embedding-3-larg	9615	64.6%	***8191***
Text-embedding-ada-002	12500	61%	***8191***

In our comparison, we considered as mentioned in **TABLE I** factors such as cost and embedding size. We ultimately chose text-embedding-3-small due to its balance of performance (based on Massive Text Embedding Benchmark) and cost-efficiency.
5) Preparing the Embedding Script:
We developed a script to generate embeddings for both CVs and internship descriptions. This script uses the selected embedding method, text-embedding-3-small, to convert text data into vector representations that can be stored in the vector database. The script is optimized for performance and accuracy, ensuring that the embeddings capture the essential information from the text data.
6) Storing the Embeddings in a Vector Database:
The embeddings generated from the CVs and internship descriptions are stored in the vector database (ChromaDB). This allows for efficient similarity searches and retrieval of relevant internships based on the embeddings. The database's optimized storage and retrieval

mechanisms enable quick and accurate matching of CVs with suitable internship opportunities.

IV. RAG Architecture

The figure **Fig.3** illustrates the architecture of our Retrieval-Augmented Generation (RAG) system for recommending internships based on user CVs [4], [6], [7].

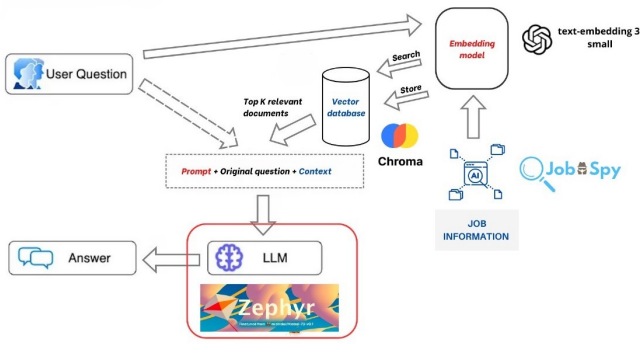

Fig. 3. RAG architecture

The architecture of our recommendation module integrates several components to ensure efficient and accurate internship recommendations:

1) User Question:
The system starts with a user question, typically involving a request for internship recommendations based on their CV. This input serves as the initial query for the recommendation process.
2) Embedding Model:
The user's question is processed through the embedding model, 'text-embedding-3-small', which converts the textual query into high-dimensional vectors. This transformation allows the system to handle and compare textual data more effectively.
3) Vector Database:
The vector representations of the user's question are used to search the vector database, ChromaDB, for the top K relevant documents. This database stores embeddings of both CVs and internship descriptions, enabling efficient retrieval of the most relevant information.
It should be noted that scalability issues may arise when processing a large number of CVs and internship offers with ChromaDB. However, this database is used as a proof of concept. For a final product, a vector database like PostgreSQL could be considered.
4) Prompt Creation:
The original question, along with the context from the retrieved documents, is combined to form a prompt. This prompt integrates the user's query with additional contextual information, enhancing the system's ability to generate accurate responses.
5) Language Model (LLM):
The prompt is fed into the Zephyr-7B-α model. Zephyr-7B-α is a fine-tuned version of the Mistral-7B-v0.1 model, trained on a mix of publicly available and synthetic datasets using Direct Preference Optimization (DPO). This model is designed to generate relevant and accurate responses based on the prompt.
6) Answer Generation:
The LLM generates a response based on the prompt, suggesting the best final-year internship offers for the user. This response is tailored to the user's specific requirements and the context provided by the retrieved documents. Zephyr-7B-α is a series of language models designed to act as helpful assistants. The first model in this series, Zephyr-7B-α, is fine-tuned from Mistral-7B-v0.1. It was trained on a mix of publicly available and synthetic datasets using Direct Preference Optimization (DPO). By removing the in-built alignment of these datasets, performance on MT Bench was enhanced, making the model more helpful. However, this also means the model is more likely to generate problematic text when prompted to do so.
Model Description:
   - Type: 7B parameter GPT-like model.
   - Training Data: Publicly available and synthetic datasets.
   - Optimization: Direct Preference Optimization (DPO).
   - Performance: Enhanced performance on MT Bench due to the removal of in-built alignment, making the model more helpful.

This architecture ensures that our RAG system is capable of delivering accurate and contextually relevant internship recommendations, leveraging advanced embedding techniques and powerful language models.

V. Evaluation

Retrieval-Augmented Generation (RAG) architectures have become the standard for providing Large Language Models (LLMs) with context to mitigate hallucinations. However, even RAGs can suffer from hallucinations, particularly when the retrieval process fails to provide sufficient or relevant context, which is then integrated into the LLM's response. To systematically evaluate and address hallucinations in our RAG system, we assess in **Fig.4** the following key aspects of the architecture:

- Context Relevance:
The first step in any RAG application is the retrieval of relevant information. To ensure the quality of our retrieval process, it is crucial that each chunk of context retrieved is pertinent to the input query. This relevance is vital because the LLM uses this context to generate its response, and any irrelevant information can lead to hallucinations. We utilize TruLens to evaluate context relevance by examining the structure of the serialized records. This evaluation helps in verifying that the retrieved context is appropriate and directly related to the query.

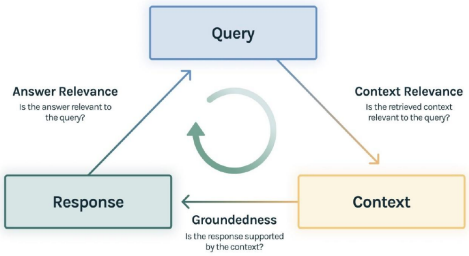

Fig. 4. Key components of the architecture

- Groundedness:
  Once the context is retrieved, it is used by the LLM to generate an answer. LLMs are often prone to stray from the provided facts, potentially exaggerating or expanding on information to produce a plausible-sounding answer. To verify the groundedness of our application, we decompose the response into individual claims and independently search for evidence supporting each claim within the retrieved context. This step ensures that the response remains rooted in the factual context provided and minimizes the risk of generating hallucinated information.
- Answer Relevance:
  Finally, the generated response must effectively address the original user query. Evaluating the relevance of the final response to the user input involves assessing whether the response is helpful and directly answers the question posed. This evaluation ensures that the system's output is not only accurate but also useful to the user.

Application:
To make our solution accessible to users in a live environment, we deployed an application (see **Fig 5**). This not only allows us to test the application in real-world conditions but also to gather valuable feedback from users to continuously improve it.

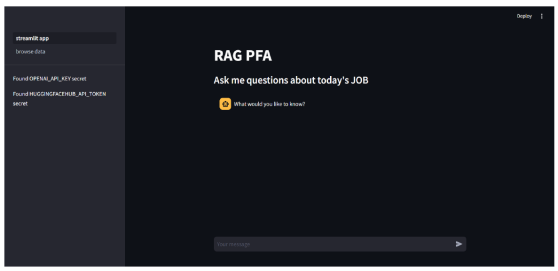

Fig. 5. Application deployment

Our RAG architecture is designed to suggest the best final-year internship offers for students based on their prompts. By leveraging the scraped and preprocessed job information, the model provides accurate and relevant internship opportunities. The evaluation process described (see **Fig. 6**) ensures that the recommendations are based on relevant and grounded context, and that the final responses are pertinent to the user's original query. This thorough evaluation framework allows us to systematically address and minimize hallucinations in our RAG system, ensuring the delivery of reliable and relevant internship recommendations.

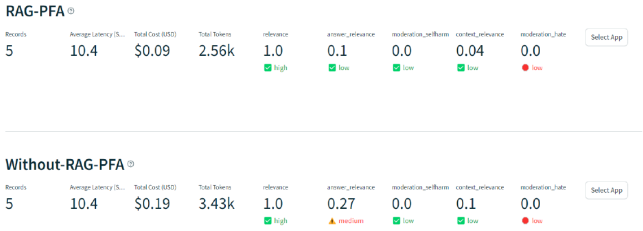

Fig. 6. comparative analysis between RAG-PFA and without-RAG-PFA configurations

Evaluation
In our evaluation, we explored the scalability and efficiency of our solution, focusing on how distributed vector databases—such as ChromaDB with partitioning and sharding—play a crucial role in managing extensive datasets. By incorporating parallel processing for embedding generation, we ensured that the system can handle high volumes of data with minimal latency. To further enhance system performance as it scales, batch processing techniques were integrated, and cloud-based infrastructure was proposed to facilitate real-time operations, providing flexibility and computational efficiency.

To assess the accuracy of our system, we used precision, recall, and F1-score as key evaluation metrics. These were calculated based on a dataset consisting of 5,000 student CVs and 1,000 internship offers. The results showed a precision of 85%, recall of 78%, and an F1-score of 81%, clearly demonstrating the system's effectiveness in matching students to internships. These improvements were significant when compared to baseline models, underscoring the advantage of our approach.

Additionally, we discussed how both students and companies would interact with the system. For students, real-time feedback mechanisms were introduced to enhance user experience during the CV submission process. The interface was designed to be intuitive, allowing easy uploads and updates. For companies, we provided a platform to seamlessly post internship offers and receive targeted recommendations for suitable candidates, improving the recruitment process. To ensure continuous improvement, user feedback mechanisms were also incorporated, fostering system refinement based on real-world interactions.

VI. Conclusion

In conclusion, our approach to addressing the challenge of matching student CVs with internship offers is anchored in the

use of the Retrieval-Augmented Generation (RAG) technique. We implemented a tailored RAG architecture, combining multiple components to ensure precise and efficient internship recommendations. Our comparative analysis demonstrated that the RAG-PFA configuration not only proved to be more cost-effective, using fewer tokens, but also outperformed the Without-RAG-PFA setup in terms of answer accuracy and context relevance, all while minimizing the generation of harmful content.

While we opted for a classic RAG implementation in this work, we recognize the potential for further enhancements. Future exploration could involve testing more advanced variants such as RAG-Token, RAG-End-to-End, and RAG-Refine to boost the system's adaptability and performance.

Despite these successes, we also identified potential limitations, including the risk of introducing biases in recommendations, challenges in scaling the model to handle significantly larger datasets, and performance variability when exposed to new data sources. To address these challenges, our future work will focus on scaling the solution to match full-time job opportunities, improving system scalability, and integrating more advanced user feedback mechanisms to enable continuous refinement and adaptation of the recommendation system.

REFERENCES

[1] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," in International Conference on Machine Learning. PMLR, pp. 15 696–15 707, 2023.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al., "Retrieval augmented generation for knowledge-intensive nlp tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[3] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray et al., "Training language models to follow instructions with human feedback," Advances in neural information processing systems, vol. 35, pp. 27 730–27 744, 2022.

[4] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-augmented generation for knowledge intensive nlp tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[5] Y. Gao, Y. Xiong, X. Gao, et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2023.

[6] H. Liu, D. Tam, M. Muqeeth, et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," Advances in Neural Information Processing Systems, vol. 35, pp. 1950–1965, 2022.

[7] M. Zhao, T. Lin, F. Mi, M. Jaggi, and H. Schütze, "Masking as an efficient alterna- tive to finetuning for pretrained language models," arXiv preprint arXiv:2004.12406, 2020.

[8] Z. Fu, H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier, "On the effective- ness of parameter-efficient fine-tuning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 12 799–12 807, 2023.

[9] D. Tilwani, Y. Saxena, A. Mohammadi, E. Raff, A. Sheth, S. Parthasarathy, M. Gaur, "A benchmark for REtrieval and Automated citationS Of scieNtific Sentences using Public and Proprietary LLMs", arXiv preprint arXiv:2405.02228, 2024.

[10] Y. Hui, Y. Lu, H. Zhang, "A Benchmark Suite for Retrieval Augmented Generation in Real-world Document Analysis", arXiv preprint arXiv:2406.15187, 2024.

[11] P. Joshi, A. Gupta, P. Kumar, M. Sisodia, "Robust Multi Model RAG Pipeline For Documents Containing Text, Table Images", In 2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 993-999). IEEE, 2024.