2024 Eighth International Conference on Parallel, Distributed and Grid Computing (PDGC) | 979-8-3315-2134-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/PDGC64653.2024.10984036

Tailored Resume Generation Using RAG with LLM as per Job Specifications

Komal Prakashchandra Patra
National College of Ireland
Dublin, Ireland
komalpatra1998@gmail.com

Manoj Diwakar
Department of CSE
Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India.
manoj.diwakar@gmail.com

Chandrakala Arya
Graphic Era Hill University,
Dehradun, India.
arya.chandrakala@gmail.com

***Abstract*—The traditional resume being common for each specific job applications which result to rejection for job seekers. These research tailors the resume based on different domain of job applications using Retrieved Augmented Generation (RAG) and LLMs. The automated system starts from resume parsing for information retrieval as per the sections using LLM model like Mixtral, Google Gemma and LLAMA-3. The resume has been further stored into vector databases such as Pinecone and MongoDB with JSON resume further retrieved using re-ranking methods such as BM25 and Cross encoder. The prompt techniques used to construct the model to make them understand better. For providing the memory to the LLM models and refining the tailored resume, the conversational buffer memory is implemented. The efficiency and the performance of the methodology is showcase through the performance evaluation metrics such as BERTscore, RAGAs Metrics and Custom metrics such as Content preservation and Job Alignment. By comparing the cosine similarity for content preservation between LLM and RAG with LLM are 72% and 89% respectively. Compared to LLM models using RAG with LLM models generates consistent and relevant tailored resumes. For a better user friendly and seamless UI, the chatbot is developed.**

***Index Terms*—Retrieval Augmented Generation (RAG), Large Language Model (LLM), Resume Generation, Conversational Memory, Prompt Engineering, Memory, LLAMA, Mixtral, BERTscore, RAGAS**

I. Introduction

Nowadays in this tough job market, surviving and getting the job has been the most tedious task. The resume of one-size-fits-all which means the generic resume methods fails to clearly present the capability of the individuals and correctly couple them with the requirements of the possible jobs. A clever way of fixing this is to have a resume that is both unique and tailored to the specific job requirements. Traditional manual resume tailoring is prone to errors and time-consuming, creating a hurdle for job seekers to present themselves effectively. So, the job seekers will miss the opportunity of presenting themselves in a better way.

The study claims that an automated solution is necessary in this case as it provides both the resume relevance and quality which results in better job search and success. Furthermore, candidates usually hit a dead end, they are in difficulty to decode the trends, understand the patterns and stick to the best practices in resume writing as an art. These issues can seriously stand in the way of a job seeker's application, thus causing disapproval due to the rejection of a job application by a hiring company. The main objective of this research is to overcome and implement the Generative AI techniques for assisting the job seekers in order to tailor their resumes for various job descriptions. This initiative is a perfect example of the urgent need for versatile, automated systems that would, in the end, ensure resumes that are relevant and up to standards. A major accomplishment like this could mean dramatic changes in the number of job seekers who would be able to find jobs more easily.

Generative AI and Natural Language Processing (NLP) have recently expanded the possibilities around automation by bringing in various domains, offering a new era of diverse applications to be automated. They include the resume scoring, pinpointing key matches and the in-depth resume evaluation anchored on specific job descriptions and enhancing the database with job recommendations tailored on individual resumes.

While developing resume automation systems using large language models (LLMs), some of the researchers experienced several difficulties. These included a lack of expertise in specific domains and the appearance of wrong information/'hallucinations' in the generated text. In contradistinction to the previous systems, the research introduces a new model that combines domain-specific expertise, focusing on the Retrieval Augmented Generation (RAG) model [1]. Furthermore, this novel approach not only eliminates the existing drawbacks but also takes advantage of open source, advanced LLM models to provide a more accurate and efficient platform for real-world applications.

II. Related Work

Existing research highlights the essential of tailoring resumes to match the job specifications using advanced large language models. The Author [2] investigated using large language models (LLMs) like GPT4 and Gemini for resume content analysis and generation. Although efficient, their cost intensive approach by using close-source LLMs model restricts the accessibility of users. However, the models lead to hallucination and maximum content length because of directly using the LLMs instead of using RAG and fine-tuning the model. The paper [3] also investigated the LLM-based tailored resume generation using the models like BARD, GPT-3.5 and LLAMA, focusing on multi-prompt techniques to enhance the contextual relevance. But the output was again led to "hallucinated" generated outputs that misinterpret skills and experience. In order to solve the security and privacy problems with synthetic resume generation, [4] employed bayesian networks alongside with the GDPR-compliance and differential privacy techniques to replicate the realistic resumes. But the major drawbacks in this study is they have simply relied on Word-Embeddings which effects the relevancy of Tailored Resume Generation. There are many researchs highlighting various studies on Resume classification, ranking, Assessing and job recommendation. One of the research [5] implemented the classification of resume using the FFNN and BERT model for generating the resume dataset. The Author [6] has crawled the Job description data from Indeed and consider the real-time human resume to enhance the Job recommendation but totally based on Semantic Similarity search. The Paper[7] presented the AIResume, which leverages the NLG and information extraction (IE) to automate the resume parsing and personalize the resume information and importantly the author have focused only on tailoring the Work history content of the resume. Further to improve the coherence and relevance, the researcher [8] has applied the BERT and GPT-2 model to fine tune the resume content while identifying the ethical issues but due to fine-tuning the GPT-2 Model with less amount of data it leads to overfitting impacting the performance. So, this can be overcome by collecting huge and diverse datasets of resume. [9] suggested using RAG with LLMs to address the hallucinations and accuracy difficulties through contextual grounding. Despite these benefits, human evaluations are subjective, indicating the future research to use objective metrics. The Paper majorly focued on generating the resume based on Word Embeddings techniques purley using NLP rather than LLM model leading to noisy context and irrelavancy generated response.

III. Materials and Methods

The tools and Techniques used to implement the automated system are below:
1. Vector Databases: Pinecone and MongoDB
2. LLM models : LLAMA-3 (for RAG) and LLAMA, Mixtral, Gemma for Resume parsing)
3. Methods: RAG, Prompt Engineering
4. Languages: Python
5. Framework: Langchain
6. Tools : ChatGroq (faster inference to LLM models), Hugging Face
7. User Interface: Streamlit

IV. Methodology

To develop an automated system that automates the tailored resume generation according to the job requirement, the author has used RAG with the open-source LLM models. The methodology shown in Fig. 1 is designed in a way which not only generates the relevant tailored resume but also can be optimized for various domains of job. The user can also further refine the tailored resume to improve from the previous resume to customize by adding the feature of a memory in a LLM.

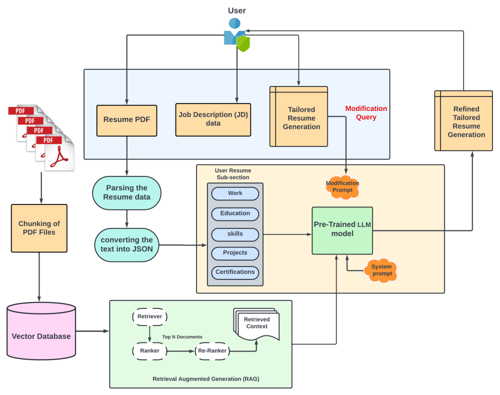


Fig. 1. Research Architecture for Tailored Resume Generation

A. Data Collection

The resumes were collected and utilised from Kaggle containing a variety of resumes consisting of different job profiles to have diverse data. The resumes are in the form of PDF containing the sub-sections such as Educations, Professionally Summary, Skills, Work Experiences, projects and Achievements. Approximately 2500+ resumes are stored into the database which does not contain the confidential information such as Names, Contacts, Profile URLs, Email Address and Home Addresses.

B. Data Processing

The text data in raw format inside the resume needs to be extracted and pre-processed to improve the efficiency. The data are organized into sub-folder, each of discrete domains. The entire PDFs directory are loaded using the langchain DirectoryLoader which concurrently extract and load the text from the PDF. Recursive Character text splitter is used to split the data into particular chunk size as the vector databases have the restriction to store the specific length of context as a vector, allowing more comprehensive and precise breakdown of the text. The most essential hyperparameter in the case of RAG's are chunk size and chunk overlap as they contribute to improve the performance of the generated text responses while maintaining the contexts between each chunks.

C. RAG

RAG acts as an external knowledge database to particular domains to narrow down the generated responses. The two essential processes in RAG are as follows: 1) retriever 2) Generation. The retriever searches the most relevant top-k documents data as per the user query. Further along with the retrieved document, the user query are then forwarded to Generation phase, which is responsible to generate the desired outcome using the LLM models [10]. Directly sending the user query to LLMs model can lead to Hallucinations because the Size of the models are vast and are trained in a wide range of public data and in such scenario, LLMs are not able to answer the domain-related queries as they are beyond training data, so making use of RAG by giving and providing latest data reduces the chances of hallucination.

The splitted chunks after the text pre-processing are converted into embedding dense vectors with the size of 1024 dimensionality. Embedding vectors are further stored into MongoDB and Pinecone vector databases. The vector database helps to store, index and query the large scale data to retrieve similar data using search techniques. Different vector databases were created to experiment with the best combination of Chunk size- Chunk overlap in order to evaluate the performance among those combinations while MongoDB has been used to store one resume PDF data file as one document. There are various distance metrics such as Cosine Similarity and Product Vector. Both distance metrics have been used for different search and ranking techniques.

The RAG has 2 ways of retrieving the metrics[1]: one is sparse-vector retrieval and the other one is dense-vector retrieval. In this research, the author has experimented with both the techniques in which BM25 has been used for sparse vector retrieval and the Cross encoder for the dense vector retrieval. The Sparse efficiently consider the Keywords and the dense vector consider the parameter of context by using the sentence embeddings where they convert the text of higher dimensions into lower dimension vector space using BERT based encoder decoder model. Hybrid search is a combination of sparse vector retrieval and dense vector retrieval where keywords and sentence embeddings are taken into consideration to improve the performance and precision of RAG systems [11].

D. Model Development

The Large language model (LLMs) are used to generate and manipulate the NLP tasks where the self-attention mechanism is used which is also referred as transformer-based architecture. There are various LLM models in which some of them are open-source such as LLAMA, Google Gemma, Mixtral, T5, etc. and some are cost-intensive but effective models such as OpenAI GPT's, Google Gemini, etc. LLM are used to perform several activities such as Question Answering (QA), NER, text generation, text-to-text generation which generate the human-like text by providing a prompt. Two most effective ways of using LLM is to either Fine-Tune or use the RAG with LLM for better optimized response. In this research, The Author has used LLM models such as LLAMA-3 for the generation while other open-source models for the resume parsing. The Large language model Meta AI (LLAMA) was developed by Meta AI. There are various previous versions of this model such as LLAMA and LLAMA2 trained on billions of parameters to understand the long-term and short-term contextual memory. Recently, the LLAMA-3 model was released by Meta AI with 2 variants pre-trained and instruction fine-tuning with 8b,70b and 450b parameters.

[1]Beyond Basic RAG: Leveraging Rerankers and Two-Stage Retrieval for Deeper Insights

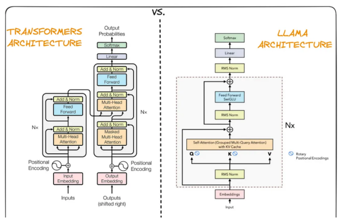


Fig. 2. LLAMA Architecture
[12]

E. Model Evaluation

As RAG enhanced the performance of LLM by downstreaming the tasks to domain-specific, it is crucial to evaluate those performance either for, retrieval process, generation process or for both as a whole RAG whole system using some evaluating techniques [13]. To check whether the outcomes are hallucinating and produce misleading or biased information,there are various frameworks/tools such as BERTScore, EvidentlyAI, DeepEval, RAGAS and ChainPool to evaluate the performance. The objective is to make sure that the RAG systems generate the answers correctly and relevant. The metrics used to measure and evaluate the performance of documents retrieved from the job description as a user query are F1 score, precision and recall. These various frameworks come under several methods as they can be embedding-based, RAG -based and LLM-based.

V. Design Specification

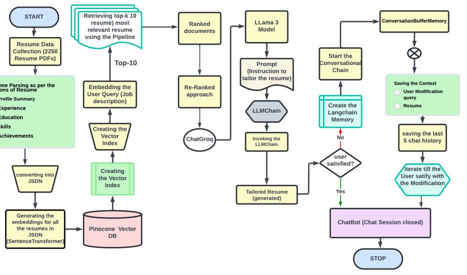

START
Resume Data Collection (2250 Resume PDFs)
Resume Parsing as per the sections of Resume
Profile Summary
Experience
Education
skills
Achievements
converting into JSON
Generating the embeddings for all the resumes in JSON (SentenceTransformer)
Pinecone Vector DB
Creating the Vector Index
Creating the Vector Index
Embedding the User Query (Job description)
Top-10
Retrieving top-k 10 resume) most relevant resume using the Pipeline
Ranked documents
Re-Ranked approach
ChatGroq
LLama 3 Model
Prompt (Instruction to tailor the resume)
LLMChain
Invoking the LLMChain
Tailored Resume (generated)
user satisfied?
No
Yes
Create the Langchain Memory
Start the Conversational Chain
ConversationBufferMemory
Saving the Context
User Modification query
Resume
saving the last 5 chat history
Iterate till the User satify with the Modification
ChatBot (Chat Session closed)
STOP

Fig. 3. System Flow Design

A. Resume Parsing

Initially the resume PDF is taken as an input from the user and is extracted using the PyPDF2 library. As shown in Figure, once the texts get extracted, the next step is to put the extracted text into sub-section of resumes such as Experience, Skills, Education,etc., using Pydantic which is the free dependency library to validate the data. To Parse the resume and segregate them into Skills correctly, the author has used the LLM models where the application system allows users to select the range of LLM models to parse and convert. Each model has their own pros and cons resulting into different generated parsed responses. Once the parsing is done using the LLM model, the resume data gives the data into JSON format with separately correct sub-section context.

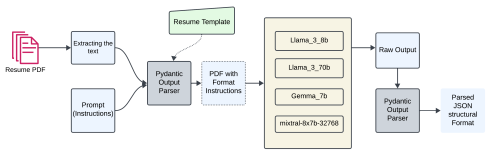


Fig. 4. Design Flow of Resume Parsing component

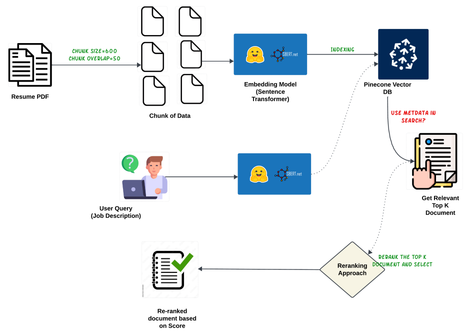

Resume PDF
CHUNK SIZE=600
CHUNK OVERLAP=50
Chunk of Data
Embedding Model (Sentence Transformer)
SBERT.net
INDEXING
Pinecone Vector DB
USE METDATA IN SEARCH?
User Query (Job Description)
SBERT.net
Get Relevant Top K Document
RERANK THE TOP K DOCUMENT AND SELECT
Reranking Approach
Re-ranked document based on Score

Fig. 5. Design Flow for RAG Retrieval component

1) Retrieval Process: As the vector data are stored into databases, the retrieval process helps to fetch and retrieve the top k relevant documents from a vector DB. The job description as a user query are converted into embedding to calculate the similarity between the vectors using cosine similarity as a distance metric. The top -k 10 resumes are fetched from Pinecone as well as the top-5 candidates are retrieved from MongoDB as they have stored one resume each as one document. The retrieved documents are assigned the scores depending on the calculation of cosine similarity. Cosine similarity is the distance metric to measure the similarity between 2 vectors from each other by finding the cosine angle between them. The cosine similarity are measured as:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}^{2} \quad (1)$$

The candidate documents are not always ranked from the top to bottom similarity score. The LLM mostly relies on top few documents due to which the performance can be harmed and irrelevant responses can be generated by LLM. So to overcome this problem, the author has used 3 re-ranking techniques to experiment further and re-score the retrieved documents to make the generative model understand the more relevant context about the job description (user query).

i) BM25: It is a deterministic model assigning a ranking to the document based on the keyword frequency, and relevantly it also contains the query relevancy. The relevant retrieved documents in the candidate set are scored using BM25. The BM25 score can be calculated as:

$$BM25 = \sum_{i}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{\text{field\_len}}{\text{avg\_field\_len}}\right)}^{3} \quad (2)$$

Using these BM25 scores, the documents are re-ordered using the BM25 scores. The documents having higher scores indicated the highest relevant document.

ii) Cross Encoder: The input for this approach are data pairs (2 retrieved documents or 2 sentences) which describes the relation and interaction between them. The encoder for the cross encoder is made up of RNN layers which encodes the information into fixed-size. In this process, the Job description (user query) is paired with each retrieved document and is further provided to the cross encoder model to understand the relevancy and context for each pair, where the model provides the score as per the relevancy between the data pairs. These scores are again re-ordered from higher to lower.

iii) Hybrid Search with Cohere API re-ranking: The Hybrid search which is a combination of both the above approaches can be performed using Cohere API re-ranking enhancing the productivity of the RAG system. Only relying on semantic-based search can lead to inefficiency and inaccurate responses. The relevancy score is obtained using the Cohere API technique.

2) Generation Process: Once the retrieval is done, the resume which is parsed into JSON format is taken as input to the LLAMA model. Along with the resume, the job description (user query) and the re-ranked retrieved documents are also taken but not directly provided to the LLM model. First, all these inputs are given to the Prompt template to create the prompt/instruction-set and organized all those inputs in a way that the model can understand to customize the resume in a structural format . Using the precise set of instructions/prompts, the LLM model provides better attention to each and every minute details to satisfy the instruction and provide the generated outcome in a better formatted way. The ChatGroq cloud is integrated with Langchain to use highly computation and heavy LLM models with billions of parameters , LLAMA through the API key. LLMchain is invoked with the sequence of processes which needs to get executed one after another to generate the outcome. For these research, the author has used stuff because it can handle the large documents which in the backend gets divided into smaller chunks. So, the Job description (user query), Resume in JSON and the retrieved document from vector databases

[2]Cosine similarity explanation

[3]The BM25 Algorithm and its variables explanation

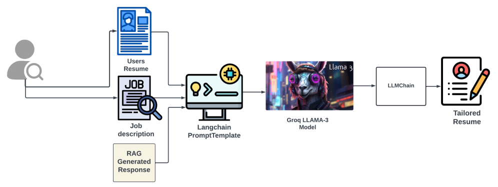


Fig. 6. Design flow for RAG Generation process

along with system and user prompts are provided to the LLM model and therefore, the resume is tailored according to the requirement of the job description.

B. *Conversational Memory*

The tailored resume satisfies the job-seeker's expectations are not mandatory and they can need further refinement to their tailored resume generation, which caused the requirement of Conversational memory between the previous chats/ tailored resume. The new refined resume should have the changes of previous tailored resume along with a new set of modifications provided by the users. Understanding these features, the author develops the interactive and smart chatbots having the conversational memory. These memories have the capability to memorize the past interaction / chat history that happened between the human users and the AI System. Langchain can use and store all those chat history for the model to get the context. The Conversational Buffer memory has been used to store and maintain the history of conversations into the buffer format.

VI. Implementation and Results

As illustrated in Figure 7, the Chatbot was developed which allowed the user to provide the job description as a user query as well as to upload the Resume in order to optimize the personalised resume as per the job specifications. The resume is parsed and converted into jSON to extract the resume information as per the Resume section. The JSON is in the structure of Pydantic information schema that is provided to the LLM model to tailor the resumes. To refine the tailored resume further, we can add the modification prompt as a user query, but in this case it directly goes to LLM model not through the RAG system as the main context has been retrieved initially. The generated resumes are in the JSON format which can be converted into a LATEX template. One of the reasons is that the ground truth for machine learning is available but not necessary in LLM as it has most complex tasks. Currently, no evaluation method is widely used or accepted because of its complexity as the results are inconsistent frequently within small changes. The author has evaluated the performance of this automated system using the embedding-based, similarity-based and LLM-based techniques. For the embedding-based evaluation technique, the BERT score is used to measure the similarity score between the generated text and reference using the embedding model. Whereas in RAGAS, the performance of the application is measured using the LLM model to calculate the metrics such as Answer relevancy, Contextual Recall, Contextual precision and Consistency, etc.

A. *Experiment 1: Using embedding Chunk-size hyper-parameter-varying metrics*

The BERT score has been used to evaluate the embedding-based metrics for both BM25 and cross encoder enhances the precision, F1 score and recall with the increase in the size of chunks followed by refinement using conversational memory. The F1 score gets enhanced up to 89% for BM25 and 87% for cross encoder. Both re-ranking techniques shoes efficient performance but BM25 takes a minor lead in gain of precision.

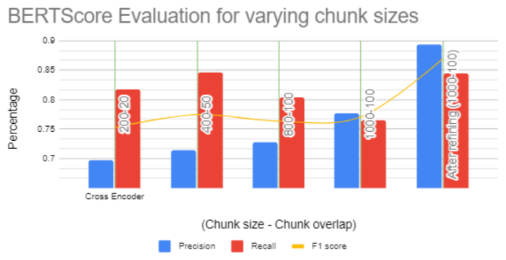


Fig. 7. Impact of BERTscore Metrics on chunk sizes for Re-Ranking technique

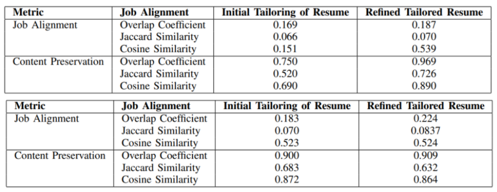
B. *Experiment 2: Using RAGAs for Chunk-size hyper-parameter-varying metrics*

Through this experiment, it has been observed that the larger chunk size allows the model to retrieve and generate the contextual relevance. In contrast, the smaller chunks allows the model to focus on specific sections to detect the minute features, enhancing precision but leads to the loss of faithfulness and coherence. Chunk size being the most crucial hyperparameter in RAG. Faithfulness in BM25's begins at 63% and decreases up to 0, while for cross encoder, it is generally low across all chunk sizes. The variability in BM25's contextual recall ranges from 25% to 50%. The evaluation in this study proves that BM25 is more reliable, especially for refining.

Fig. 8. Impact of chunk size on RAGAs Metrics

C. *Experiment 3: Evaluating using Job Alignment and Content Preservation*

Evaluating the similarity between the token an the latent spaces, the job alignment calculates how job description is close/similar to resumes, while the content preservation feature is responsible to assess the system's capability to maintain the content of the user's original resume in the tailored resume. This metrics provides how well the tailored resume keeps the context and relevance from the original one. Under this two main approach, the author has calculated the overlap coefficient and the jaccard similarity for each one of them. Analyzed and compared the overlap keywords from two different sets utilizing the overlap coefficient to high accuracy, particularly emphasizing on evaluating smaller sets. By using the algorithm of Jaccard similarity, it can be determined how similar two sets are when they intersect with each other with a scale from 0 (no similarity) to 1 (Full similarity). Demonstrated enhancement in all evaluated similarity metrics (Job alignment and content preservation) through BM25, particularly noted in overlap coefficient and cosine similarity. In Fig. 9, the evaluation score for the Job alignment and content preservation is provided.



Metric	Job Alignment	Initial Tailoring of Resume	Refined Tailored Resume
Job Alignment	Overlap Coefficient	0.169	0.187
	Jaccard Similarity	0.066	0.070
	Cosine Similarity	0.151	0.539
Content Preservation	Overlap Coefficient	0.750	0.969
	Jaccard Similarity	0.520	0.726
	Cosine Similarity	0.690	0.890

Metric	Job Alignment	Initial Tailoring of Resume	Refined Tailored Resume
Job Alignment	Overlap Coefficient	0.183	0.224
	Jaccard Similarity	0.070	0.0837
	Cosine Similarity	0.523	0.524
Content Preservation	Overlap Coefficient	0.900	0.909
	Jaccard Similarity	0.683	0.632
	Cosine Similarity	0.872	0.864

Fig. 9. Comparison of BM25 and Cross Encoder across different stages and Custom Metrics

Since the score of hallucination was around 0.1890 indication the very low hallucinations means the model generating the tailored is considered to be relevant, accurate and rooted to ground truth.

VII. Conclusion and Future Work

The research focuses on creating tailored resumes based on different job specifications to address the issues encountered by job seekers and meet the objectives. The Resume dataset PDFs contains the resumes from a variety of domains to increase the probability of finding and retrieving the similar documents with the highest ranking score. The original resume extracted the information as per the section using the Large language models and converted into JSON. The llama-3-70b model works best in the case of information retrieval compared to other open source models. For tailoring the resume, the RAG was implemented with LLM by experimenting with different retrieval and re-ranking techniques where BM25 which is sparse vector retrieval were able to generate the response better.Further the gap can be fulfilled in future by focusing on the evaluation of this automated system using the reliable evaluation method. Additionally, instead of RAG, it can be explored by implementing the RAFT (which is a mixture of Fine-tuning and RAG) or Knowledge graph to improve the performance of this system.

References

[1] 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.

[2] S. B. Zinjad, A. Bhattacharjee, A. Bhilegaonkar, and H. Liu, "Resumeflow: An llm-facilitated pipeline for personalized resume generation and refinement," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 2781–2785.

[3] R. J. Sunico, S. Pachchigar, V. Kumar, I. Shah, J. Wang, and I. Song, "Resume building application based on llm (large language model)," in *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2023, pp. 486–492.

[4] A. Bruera, F. Alda, and F. Di Cerbo, "Generating realistic synthetic curricula vitae for machine learning applications under differential privacy," in *Proceedings of the Workshop on Ethical and Legal Issues in Human Language Technologies and Multilingual De-Identification of Sensitive Data In Language Resources within the 13th Language Resources and Evaluation Conference*, I. Siegert, M. Rigault, and V. Arranz, Eds. Marseille, France: European Language Resources Association, Jun. 2022, pp. 53–63. [Online]. Available: https://aclanthology.org/2022.legal-1.11

[5] P. Skondras, P. Zervas, and G. Tzimas, "Generating synthetic resume data with large language models for enhanced job description classification," *Future Internet*, vol. 15, no. 11, 2023. [Online]. Available: https://www.mdpi.com/1999-5903/15/11/363

[6] S. Guo, F. Alamudun, and T. Hammond, "Résumatcher: A personalized résumé-job matching system," *Expert Systems with Applications*, vol. 60, pp. 169–182, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416301798

[7] J. Balaji, M. Sigdel, P. Hoang, M. Liu, and M. Korayem, "Airesume: Automated generation of resume work history." in *KaRS@ CIKM*, 2019, pp. 19–25.

[8] S. S. Kale and W. B. Andreopoulos, "Job tailored resume content generation," in *2023 IEEE Ninth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, 2023, pp. 40–47.

[9] P. F. Foulds, R. James, and S. Pan, "Ragged edges: The double-edged sword of retrieval-augmented chatbots," *arXiv preprint arXiv:2403.01193*, 2024.

[10] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-augmented generation for ai-generated content: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2402.19473

[11] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," 2022. [Online]. Available: https://arxiv.org/abs/2202.01110

[12] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[13] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, "Evaluation of retrieval-augmented generation: A survey," *arXiv preprint arXiv:2405.07437*, 2024.