Resume Screening Using Large Language Models

Esmail Salakar
Department of Computer Engineering
K. J. Somaiya Institute of Technology
Mumbai, India
esmail.s@somaiya.edu

Jivitesh Rai
Department of Computer Engineering
K. J. Somaiya Institute of Technology
Mumbai, India
jivitesh.r@somaiya.edu

Aayush Salian
Department of Computer Engineering
K. J. Somaiya Institute of Technology
Mumbai, India
aayush.salian@somaiya.edu

Yasha Shah
Department of Computer Engineering
K. J. Somaiya Institute of Technology
Mumbai, India
yasha.shah@somaiya.edu

Dr. Jyoti Wadmare
Department of Computer Engineering
K. J. Somaiya Institute of Technology
Mumbai, India
jyoti@somaiya.edu

2023 6th International Conference on Advances in Science and Technology (ICAST) | 979-8-3503-5981-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAST59062.2023.10454984

Abstract - **In today's competitive job market, HR professionals face a significant challenge due to the overwhelming resumes received for job openings from numerous candidates. This paper attempts to provide an automated solution using one of the latest developments in ML technologies, Large Language Models (LLMs). LLMs have come into the spotlight in recent years, largely because of the introduction of Artificial Intelligence systems. The aim of this paper is to research the application of LLMs, so as to deliver a solution to the challenges faced by recruiters, thus simplifying the procedure while increasing efficiency.**

Keywords— Large Language Model, FIASS Algorithm, Vectorization, HNSW Algorithm, CNN, Euclidean Distance

I. INTRODUCTION

In today's dynamic and competitive job market, resume screening remains an important and challenging task for companies trying to identify the best candidates for their vacancies. This process involves sifting through a vast array of resumes, each demonstrating unique qualifications, experience, and skills. Conventional resume review techniques that rely on manual assessment and keyword matching are getting more and more laborious, inefficient, and prone to bias.

The advent of language models (LM) ushered in a new era of natural language processing, providing innovative solutions to address the challenges associated with resume screening. The Language Model for Resume Review (LMRS) is one of these LMs that has gained popularity as a potent tool for streamlining and enhancing the resume screening procedure. Often referred to as the Large Language Model (LLM), this technology transforms the way businesses find, assess, and choose candidates by utilising the capabilities of artificial intelligence and machine learning.

This research paper delves into the field of resume screening using LLMs and explores the potential of this cutting-edge technology to revolutionize and streamline the hiring process. We aim to provide a comprehensive picture of the current resume review landscape by understanding the opportunities and limitations of LLMs and related ethical considerations.

II. LITERATURE SURVEY

LLMs are prepared on enormous datasets of content and code, and they are able to memorize complex designs in dialect. This makes them well-suited for an assortment of NLP assignments, counting content generation, interpretation, and question-answering.

Table 1 summarizes ten research papers on a variety of topics, including 10 papers. The table provides a concise overview of each paper's main findings and conclusions, highlighting the key contributions of each study to the field.

Large Language Models (LLMs) are neural network-based natural language processing models, predominantly built on transformer architectures. They go through a two-step preparation process: pre-training on limitless content datasets, where they pick up dialect structures and knowledge, and fine-tuning on domain-specific data to adapt to particular assignments. By breaking down text into smaller units and employing self-attention mechanisms, LLMs can recognise long-range dependencies in language and understand context. These models can carry out tasks like sentiment analysis, text generation, and translation because they have a large number of parameters—often hundreds of millions or billions—that they learn over training.

Witteveen et al., introduces a new technique for paraphrasing text using LLMs. The authors show that their approach can generate high-quality paraphrases at both the sentence and paragraph levels. This suggests that LLMs can be used for a variety of text generation tasks, in addition to translation and question answering. [12]

Sarkar et al., presents a study that investigates the experiences of programmers who use LLMs. The authors find that LLMs can be helpful for tasks such as code generation, autocompletion, and debugging. However, they also find that LLMs can be unreliable and can sometimes generate incorrect or incomplete code. [13]

TABLE I. Comparative study of previous research papers.

Aspect	**Resume Screening using NLP and LSTM**	**Applicant Screening System**	**Automated Resume Screener**	**Web Application for Screening Resume**	**Resume Classification Using ML Techniques**	**Resume Screening and Ranking using CNN**	**Screening and Ranking Resumes using Stacked Model**	**Resume Parse based on Multi-label Classification using NN models**	**Resume Classification using Elite Bag-of-Words Approach**	**Resume Ranking based on Job Description using NER model**
Technology Used	NLP, LSTM	NLP, Cosine Similarity	NLP, Machine Learning	NLP, Machine Learning	NLP, Named Entity Recognition, ML Models	CNN, LSTM, ML Models	ML Models	Neural Networks	Elite Bag-of-words	ML Models,NLP,NER
Main Focus	Automated Resume Screening	Compatibility Calculation	Automated Resume Screening	Automated Resume Screenings	Resume Classification	Resume Screening and Ranking	Resume screening and ranking	Resume classification	Resume Classification	Resume Ranking
Key Features	Efficiency, Accuracy, Customization	Compatibility Assessment	Efficiency, Format	Automation, Efficiency	Classification Models	Automation	Stacked models	Comparison of models	Improving accuracy of NLP techniques	Automation
Notable Methods/ Tools Mentioned	ATS, Linear SVC, K-Nearest Neighbor	NLTK, spaCy, Cosine Similarity	AI-based Keyword Matching	Semi-Supervised Learning	NLP, Named Entity Recognition, ML Models	LSTM,CNN	XGBoost	CRNN, BiLSTM	Elite bag-of-words vectorization	SpaCy NER model
Best-Performing Model and Accuracy (if any)	Accuracy-87%	Not specified	Accuracy-87%, Precision-83%, Recall-85%	Not specified	Random Forest (91.38%)	CNN(99.48%)	Stacked model(83%)	CRNN(96.41)	Elite Keywords(62.6%)	F1 – 84.84%, Prec – 86.92%, Recall – 90.61%

Liu et al., investigates how LLMs use long contexts. The authors find that LLMs are not able to fully utilize long contexts, and their performance decreases as the context length increases. This suggests that new methods are needed to help LLMs process long contexts more effectively [8].
Caines et al., explores the potential of LLMs to be used in language teaching and assessment technology. The authors discuss the potential benefits and challenges of using LLMs in these applications. They also propose a number of research directions for the future. [9]
Wang et al., proposes a new approach to pruning LLMs called FLOP (Factorized Low-rank Pruning). FLOP is able to achieve significant compression levels while maintaining or even enhancing model performance. This makes it a valuable tool for deploying LLMs in practical applications. [12]
Shinn et al., introduces a new LLM called Reflexion, which is able to learn and perform tasks autonomously using self-reflection. Reflexion is able to improve its performance over time by reflecting on its own experiences and learning from its mistakes. [13]

After extensive research about various approaches made in the following research papers as seen Table 1 shows that machine learning and deep learning using neural networks can be observed with greater accuracy than accuracy based on matching strings, which is less likely to be accurate. Comparative study is shown in table I.

III. METHODOLOGY

Overview of the system: Fig. 1 shows the representation of the processes to be followed by our system to generate and output with different queries.

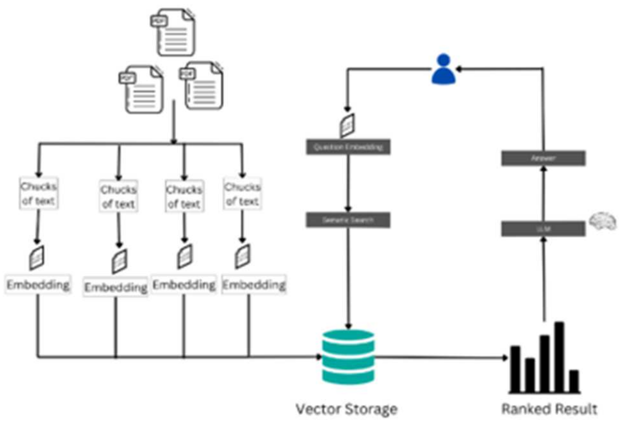


Fig. 1. Overview of the system

A. DATA PREPROCESSING

Data preprocessing commences with the conversion of text from PDF files into plaintext. This may be accomplished via a PDF parser, a software application that extracts text, images, and other data from PDF files.

The subsequent step in the document processing and ranking process entails the transformation of the extracted text into data chunks. The chunking strategy determines how documents are partitioned into smaller segments for processing by a machine learning model. There are two primary approaches to chunking: storing entire documents or storing them in smaller sections.

Storing entire documents is the most straightforward approach, but it can be inefficient for large documents, as the entire document must be processed every time a query is made. On the other hand, storing documents in smaller chunks is more efficient, as only the relevant sections need to be processed when a query is initiated. It is crucial to select an appropriate chunk size, as excessively small chunks can result in information loss, while overly large chunks can be inefficient.

For resumes that are only one to two pages long, it may be advantageous to use the entire resume document as a single chunk for an LLM. This preserves the context of the resume and allows for more complex and nuanced analysis. This can be useful for research on topics such as career trajectories, skill development, and job satisfaction. Additionally, it can be more efficient for certain types of research, such as identifying keywords or phrases that are common to successful candidates.

B. DATA TRANSFORMATION

Tokenization is parsing a text into its constituent elements, such as words, phrases, or sentences. This is essential for many natural languages processing tasks, including vector embedding.

Vector embedding is a type of representation that maps words or other text fragments to vectors of numbers. These vectors capture the semantic meaning of the words or text fragments. Vector embedding is essential for making resume data accessible and usable in machine learning models. By converting textual information into numerical format, vector embedding enables model processing and boosts model accuracy. Additionally, vector embedding facilitates resume comparison, enabling the identification of well-suited candidates for specific job roles through similarity calculations.

There are two main types of vector embeddings: dense vectors and sparse vectors. Dense vectors are vectors that have all of their elements set to a number. This means that they can represent any possible word or text fragment, without any redundancy. Sparse vectors are vectors that have only a few of their elements set to a number. This means that they are more efficient to store and use, but they may not be able to represent all possible words or text fragments.

One approach to creating vector embeddings shown in Fig. 2 is feature engineering. This approach requires domain knowledge and can be expensive to scale. An alternative is to train a deep neural network to translate resume data into vector embeddings. Deep neural networks can learn the semantic meaning of resume data directly, without the need for domain knowledge. The resulting vector embeddings are high-dimensional and dense, which allows for the development of more accurate and efficient resume screening and ranking systems.

C. VECTOR STORAGE

Vector embeddings are high-dimensional data structures generated by AI models, such as large language models. They have a vast number of features, making them computationally expensive to store and query.

Vector databases are specialized databases designed to efficiently store and query vector embeddings. They offer capabilities that are not available in standalone vector indexes, such as the features of a traditional database. Conventional scalar-based databases are unable to handle the complexity and scale of vector embeddings. This makes it difficult to extract insights and perform real-time analysis. Vector databases offer the performance, scalability, and flexibility required to fully leverage vector embeddings. They can perform advanced functions such as semantic information retrieval, long-term memory, and more. Using vector databases, we can search for resumes that are semantically similar to a given query, or implement long-term memory.

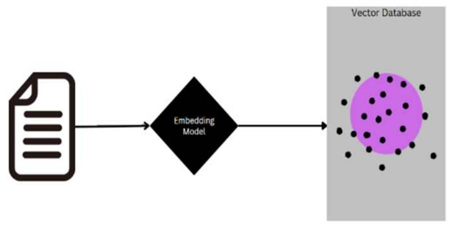


Fig. 2. Conversion of data

D. VECTOR SEARCH

Vector search is a technique used in information retrieval and data analysis to find and retrieve similar items from a dataset based on their vector representations. Here, a vector represents an item's features or attributes as numerical values.

1) Distance and Similarity Metrics

- Remove measurements, such as L1 Manhattan separate and L2 Euclidean separate, degree the disparity between vectors.
- Cosine similarity measures the similarity between vectors.

Higher distance metric values indicate less similarity, while higher similarity metric values indicate more similarity.

2) Indexing

Multiple algorithms are available to support the establishment of a vector index, all aimed at enabling swift queries by constructing a data structure conducive to rapid traversal. Typically, these algorithms condense the original vector representation into a compressed format to enhance query optimization.

a) Product Quantization for Memory Reduction

- Dense embedding vectors can be memory-intensive.
- Product quantization (PQ) is a method that reduces the memory usage by compressing vectors.
- Quantization involves representing vectors using a smaller set of vectors.

b) FAISS Algorithm

- FAISS stands for Facebook AI Similarity Search.
- It is a clustering algorithm that computes L2 Euclidean distance between query vectors and all other points.
- To optimize the search process, FAISS uses Voronoi cells to identify the closest centroid to the query vector.
- It uses an approximation to compute the closest centroid as against KNN which uses brute force by computing distance with all centroids.

c) HNSW Algorithm

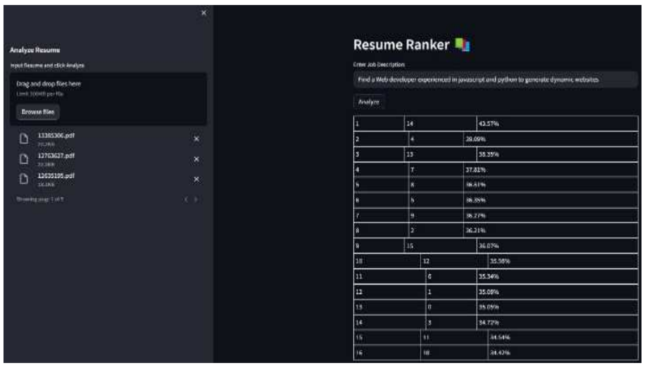
- HNSW stands for Hierarchical Navigable Small Worlds.
- It is a proximity graph-based approach that uses Euclidean distance as a metric.
- HNSW utilizes a linked list or skip list structure and introduces hierarchy to efficiently navigate the graph and find nearest neighbours.

IV. RESULTS

To Generate /Implement the results produced by our process in a very small-scale environment to check its efficiency and produce conclusions.

For this we have taken 135 pdfs of various different IT professionals with different skill sets and used a prototype LLM software Chromadb. The Query passed on to the software was as follow “Find a Web developer experienced in JavaScript and python to generate dynamic websites” as you see the software then generates vector indexes for the the query itself as well as the pdfs that has been uploaded on it. After matching the indexes, a ranking is generated to showcase the most suitable candidate.

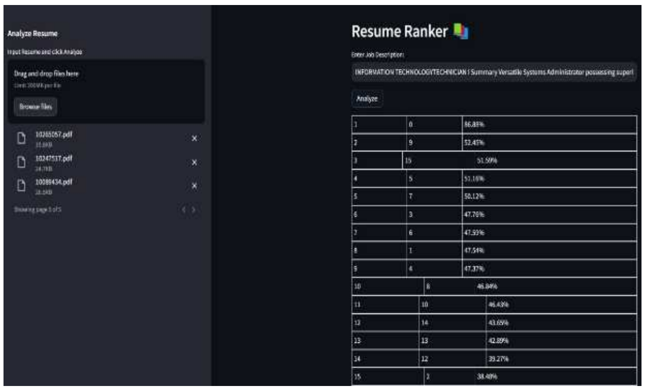
As it can be observed from the Fig. 3, candidate no. 15 seems to be the best fit for the given job description.



Find a Web developer experienced in javascript and python to generate dynamic websites

1	14	43.57%
2	4	39.09%
3	13	38.35%
4	7	37.81%
5	8	36.61%
6	5	36.35%
7	9	36.27%
8	2	36.21%
9	15	36.07%
10	12	35.38%
11	6	35.34%
12	1	35.08%
13	0	35.05%
14	3	34.72%
15	11	34.54%
16	10	34.42%

Fig. 3. Results with Detailed Description



INFORMATION TECHNOLOGYTECHNICIAN I Summary Versatile Systems Administrator possessing superl

1	0	86.83%
2	9	52.45%
3	15	51.50%
4	5	51.16%
5	7	50.12%
6	3	47.76%
7	6	47.59%
8	1	47.54%
9	4	47.37%
10	8	46.84%
11	10	46.43%
12	14	43.69%
13	13	42.89%
14	12	39.27%
15	2	38.48%

Fig. 4. Results with Extensive Description

To ensure this is just not a simple string-matching software we also copy all the contents of the pdf and run the analyser again showcasing the vector indexing observed in the Fig. 4.

V. CONCLUSION

The merging of vector embeddings and large language models (LLMs) has the potential to transform resume screening and ranking. It is possible to train LLMs to create vector embeddings of resumes that capture the semantic subtleties of the text, including the applicant's credentials, experience, and abilities. Then, by utilising these vector embeddings, a multitude of advanced tasks can be carried out, including the recognition of semantically similar resumes, the implementation of long-term memory, the creation of personalised ranking systems, and the creation of innovative matching and search algorithms.

REFERENCES

[1] S. Mhatre, B. Dakhare, V. Ankolekar, N. Chogale, R. Navghane and P. Gotarne, "Resume Screening and Ranking using Convolutional Neural Network," 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2023, pp. 412-419, doi: 10.1109/ICSCSS57650.2023.10169716.

[2] Jiahao Liu, Yifan Shen, Yijie Zhang, and Sujatha krishnamoorthy. 2021. Resume Parsing based on Multi-label Classification using Neural Network models. In Proceedings of the 6th International Conference on Big Data and Computing (ICBDC '21). Association for Computing Machinery, New York, NY, USA, 177–185.https://doi-org.library.somaiya.edu/10.1145/3469968

[3] M. Sharma, G. Choudhary and S. Susan, "Resume Classification using Elite Bag-of-Words Approach," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 1409-1413, doi: 10.1109/ICSSIT55814.2023.10061036.

[4] Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. "Large language models in machine translation." (2007).

[5] Witteveen, Sam, and Martin Andrews. "Paraphrasing with large language models." *arXiv preprint arXiv:1911.09661* (2019).

[6] Sarkar, Advait, Andrew D. Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, and Ben Zorn. "What is it like to program with artificial intelligence?." *arXiv preprint arXiv:2208.06213* (2022)

[7] Sun, Haitian, William W. Cohen, and Ruslan Salakhutdinov. "Answering Ambiguous Questions with a Database of Questions, Answers, and Revisions." *arXiv preprint arXiv:2308.08661* (2023).

[8] Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. "Lost in the middle: How language models use long contexts." *arXiv preprint arXiv:2307.03172* (2023).

[9] Caines, Andrew, Luca Benedetto, Shiva Taslimipoor, Christopher Davis, Yuan Gao, Oeistein Andersen, Zheng Yuan et al. "On the application of Large Language Models for language teaching and assessment technology." *arXiv preprint arXiv:2307.08393* (2023).

[10] Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. "A survey of large language models." *arXiv preprint arXiv:2303.18223* (2023).

[11] S Marcondes, F., Almeida, J. J., & Novais, P. (2023). Large Language Models: Compilers for the 4^{th} Generation of Programming Languages?(Short Paper). In *12th Symposium on Languages, Applications and Technologies (SLATE 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[12] Wang, Ziheng, Jeremy Wohlwend, and Tao Lei. "Structured pruning of large language models." *arXiv preprint arXiv:1910.04732* (2019).

[13] Shinn, Noah, Beck Labash, and Ashwin Gopinath. "Reflexion: an autonomous agent with dynamic memory and self-reflection." *arXiv preprint arXiv:2303.11366* (2023).

[14] Bellan, Patrizio, Han van der Aa, Mauro Dragoni, Chiara Ghidini, and Simone Paolo Ponzetto. "PET: An Annotated Dataset for Process Extraction from Natural Language Text." *arXiv preprint arXiv:2203.04860* (2022).

[15] P. M. A. Kumar, E. Pugazhendhi and K. V. Lakshmi, "Cloud Data Storage Optimization by Using Novel De-Duplication Technique," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 436-442, doi: 10.1109/ICSSIT53264.2022.9716508.

[16] P. Abedin, O. Chubet, D. Gibney and S. V. Thankachan, "Contextual Pattern Matching in Less Space," 2023 Data Compression Conference (DCC), Snowbird, UT, USA, 2023, pp. 160-167, doi: 10.1109/DCC55655.2023.00024.

[17] Bharti, S. K., & Babu, K. S. (2017). Automatic keyword extraction for text summarization: A survey. *arXiv preprint arXiv:1704.03242*.

[18] Ibrahim, N., Hanum, H. M., Bakar, Z. A., & Abdullah, N. A. S. (2021, June). Student-Industry Matching for Internship Placement. In *2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP)* (pp. 122-126). IEEE.

[19] Wen-jian, W., & Shun-xiang, W. (2009, July). A jumping string mode matching algorithm. In *2009 4th International Conference on Computer Science & Education* (pp. 1181-1185). IEEE.