*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 2, Ser. 5 (Mar. – Apr. 2025), PP 01-10*
*www.iosrjournals.org*

# Comparison Of Models For Resume-JD Matching: BERT, Gemini, And Llama 3.1

Sarah Khan, Amisha Shahi, Altaf Alam, Abhigyan Bafna, Dr. Shanthi Therese
*Information Technology/Mumbai University, India*

***Abstract:***
*Matching resumes to job descriptions (JDs) is a critical process in personalized job search platforms, ensuring that candidates are effectively aligned with suitable opportunities. This study evaluates three key models—BERT, Gemini, and LLaMA 3.1—focusing on their capabilities, limitations, and comparative performance in addressing challenges such as semantic matching, domain-specific context handling, and variability in text representation. Comprehensive testing reveals insights into each model's strengths and weaknesses, offering a clear roadmap for selecting the optimal solution for AI-driven recruitment platforms. Our findings underscore LLaMA 3.1's superiority, highlighting its advanced contextual understanding, robust domain-specific matching capabilities, and ability to handle ambiguous resumes, establishing it as the most effective model for this application.*
***Key Word****: Resume, Job Description, BERT, Gemini, LLaMA.*

---

Date of Submission: 10-04-2025 Date of Acceptance: 20-04-2025

---

## I. Introduction

Effective job matching is a cornerstone of personalized recruitment platforms. AI-based approaches leverage natural language processing (NLP) to align resumes with job descriptions (JDs), overcoming challenges of variability in language, semantics, and domain-specific terminologies. [1][7] Traditional job-matching methods struggle with context sensitivity, semantic understanding, and ambiguity, often leading to poor candidate-job alignment.[9] Advanced NLP models like BERT, Gemini, and LLaMA 3.1 promise improvements but require evaluation for practical applications.[2]-[4]

This study aims to:
1. Evaluate the performance of BERT, Gemini, and LLaMA 3.1 in resume-JD matching.
2. Identify strengths and limitations of each model.
3. Recommend the most effective model for AI-driven recruitment platforms.

The research focuses on the technology domain, using real-world datasets to evaluate model performance in semantic matching, context sensitivity, and domain-specific adaptability.

## II. Literature Review

Traditional methods for resume-to-job description (JD) matching have relied on structured and unstructured approaches. The evolution of these techniques highlights the ongoing advancements in natural language processing (NLP) and artificial intelligence (AI), leading to more accurate and efficient hiring processes.[10][11]

### Traditional Methods for Resume-JD Matching

#### Rule-Based Systems

Early systems for resume matching were primarily rule-based, focusing on exact keyword matching between resumes and job descriptions. These systems followed predefined sets of rules and heuristics, often utilizing Boolean logic to filter candidates. While they provided a straightforward method for candidate shortlisting, they exhibited several shortcomings:
- Lack of flexibility in handling variations in job titles and skills.
- Inability to recognize synonyms or context, leading to false negatives.
- Heavy dependence on manually curated dictionaries, requiring frequent updates to remain relevant. [13]

Despite these drawbacks, rule-based approaches served as the foundation for more sophisticated techniques and are still used in conjunction with modern AI-driven methods.

**Vector Space Models**

The introduction of Vector Space Models (VSM) represented a significant advancement over rule-based methods. These models transformed textual data into numerical representations, allowing similarity computations between resumes and job descriptions. The key advantages included:

● Representation of documents as vectors in a multidimensional space.
● Ability to calculate cosine similarity for assessing document compatibility.

However, VSM struggled with certain linguistic challenges:

● It treated words as independent entities, disregarding semantic relationships.
● It failed to account for synonymy and polysemy, leading to suboptimal matching results.
● The reliance on term frequency-inverse document frequency (TF-IDF) limited contextual understanding. [14]

Despite these limitations, VSM laid the groundwork for embedding-based approaches that improved semantic representation.

**Traditional Methods Utilizing Doc2Vec**

One historical approach that attempted to enhance document representation was **Doc2Vec**, an extension of Word2Vec designed to generate vector embeddings for entire documents.[14] Doc2Vec facilitated resume-JD comparison through the following process:

1. Training the model on large datasets to learn document representations.
2. Generating fixed-length embeddings for resumes and job descriptions.
3. Computing cosine similarity between vectors to determine relevance.

Doc2Vec improved upon VSM by capturing semantic relationships between words and phrases, resulting in better text comparisons. However, it had its own set of challenges:

● Limited domain adaptability, requiring retraining for different industries.
● Difficulty in handling contextual variations, particularly for ambiguous job titles and skill descriptions.
● Heavy reliance on labeled data for training, making large-scale deployment costly.

Recruitment platforms utilizing Doc2Vec typically followed a workflow where candidates with high similarity scores were shortlisted, allowing hiring managers to focus on the most relevant applications. [12]

**Advanced Information Extraction and Matching Techniques**

More recent advancements in NLP have introduced sophisticated methods for extracting information and improving resume-JD matching. [1][7][8][14] These include:

**Information Retrieval Using Rule-Based Matchers**

Modern rule-based matchers leverage NLP libraries such as **Spacy PhraseMatcher** to extract structured information (e.g., education, skills, experience) from job descriptions. [7] These systems create dictionaries of relevant categories, enabling precise entity recognition. Some key features include:

● Identification of domain-specific terms through curated knowledge bases.
● Rule-based phrase matching for structured extraction.
● Enhanced entity recognition by integrating machine learning-based classifiers.

While these approaches improve accuracy, they still rely on predefined lexicons, making them less adaptable to evolving job market trends.

**Semantic Matching Using Pre-Trained Models**

The adoption of **pre-trained transformer-based models** has revolutionized resume matching. [15][16] Unlike keyword-based systems, these models capture semantic meaning and contextual nuances. Prominent models include:

● all-mpnet-base-v2
● paraphrase-MiniLM-L6-v2
● all-roberta-large-v1

These models generate embeddings for textual data, enabling effective similarity computation via cosine similarity. When applied to resume-JD matching, they outperform traditional methods by:

● Recognizing synonyms and related phrases (e.g., “software developer” and “programmer”).
● Capturing context-dependent meanings (e.g., “lead engineer” versus “engineering lead”).
● Handling domain-specific terminologies with greater accuracy.

Studies have shown that semantic embeddings significantly enhance match precision, leading to better hiring outcomes.

**Evolution of NLP in Recruitment Platforms**

The introduction of **transformer-based models** such as **BERT** marked a paradigm shift in NLP for recruitment.[17][18] Unlike previous models, BERT's bidirectional training allowed it to understand context more effectively. However, it faced certain challenges:

- **Input length limitations**, restricting its ability to process long resumes and job descriptions.
- **Need for domain-specific fine-tuning**, requiring large labeled datasets for optimal performance.

To address these limitations, newer models such as **Gemini** and **LLaMA 3.1** have emerged [19]

**Overview of Existing Models and Prior Research**

**BERT (Bidirectional Encoder Representations from Transformers)**

BERT excels in general NLP tasks due to its ability to understand bidirectional context. However, for recruitment applications, extensive fine-tuning is needed to adapt it to industry-specific terminology.

**Gemini**

Designed for lightweight text matching, Gemini offers efficient processing but struggles with complex contextual nuances. While it provides faster inference times, its lack of deep contextual analysis limits its performance in high-stakes recruitment scenarios.

**LLaMA 3.1**

A more advanced architecture, LLaMA 3.1 is specifically tailored for **long-form text processing** and **semantic inference**. It provides superior accuracy in resume-JD matching by:

- Capturing deeper contextual meanings.
- Handling longer text inputs effectively.
- Reducing bias through improved pre-training methodologies.

## III. Dataset Description

To evaluate the performance of BERT, Gemini, and LLaMA 3.1 in Resume-JD matching, we conducted controlled experiments using a standard resume and job description as test inputs. The resume and job description were carefully selected to include diverse elements such as domain-specific terminology, varied formats, and ambiguous role descriptions to test the models' adaptability and contextual understanding.

For our research, we have compiled a dataset consisting of resumes and corresponding job descriptions to evaluate their compatibility using **BERT, Gemini, and LLaMA 3.1**.

**Data Sources**

1. The **resume dataset** has been sourced from Kaggle. It contains a diverse collection of resumes across multiple industries and job roles.
2.The **job descriptions dataset** has been **created manually** by curating real-world job postings from various sources, ensuring relevance and diversity.

**Final Merged Dataset**

After merging the Kaggle resume dataset with our manually curated job descriptions,the final dataset consists of:

**1.Total Records: 4966**

**2.Total Columns: 5**

| Column Name | Description | Data Type |
|---|---|---|
| jd_content | Textual Representation of job description | String |
| type | Type/category of the job | String |
| role | Specific job role | String |
| resume_content | Textual representation of the resume | String |
| category | Broad category under which the job falls | String |

**Table.1** Kaggle Dataset With Job Descriptions

**Dataset Structure**

- Resumes are provided in textual format and categorized into specific job fields such as IT, Finance, Healthcare, and more.

- Job descriptions are structured to match real-world hiring practices and include key responsibilities, required skills, and qualifications.
- The dataset enables training and evaluation of models based on how well a resume aligns with a given job description.

By combining real resumes from Kaggle with a manually curated job description dataset, we ensure that our dataset is both diverse and representative of real-world hiring scenarios

## IV. Methodology

We evaluated the models using a curated dataset of resumes and job descriptions from various industries, emphasizing contextual and semantic matching. Key performance indicators included:

- Contextual sensitivity
- Handling of abbreviations and technical jargon
- Long-form text processing
- Domain-specific adaptability
- Synonym and variant matching
- Handling of ambiguous resumes

**Observations and Analysis**

**BERT: Limitations in Contextual Sensitivity**

BERT (Bidirectional Encoder Representations from Transformers) has revolutionized natural language processing (NLP) with its ability to understand bidirectional context. However, several limitations hinder its performance in specific scenarios:

Limited Contextual Understanding:

BERT struggles with specialized contexts, such as abbreviations or niche technical terminologies. For instance, it may fail to fully understand domain-specific acronyms like "MERN" (MongoDB, Express.js, React.js, Node.js) without additional fine-tuning. This can result in suboptimal outputs when processing resumes, job descriptions, or other texts containing industry-specific jargon.

Shallow Semantic Matching:

While BERT is adept at recognizing surface-level keyword matches, it often lacks depth in semantic understanding. For example, it may treat “API development” and “built REST services” as unrelated terms, despite their close equivalence in meaning. This limitation affects tasks requiring nuanced semantic alignment, such as resume parsing or job recommendation systems.

Poor Handling of Ambiguity:

Ambiguities in text, such as resumes listing multiple roles or hybrid skills, can confuse BERT. For example, a profile mentioning experience in both "software engineering" and "technical project management" may lead to inconsistent interpretations, impacting the relevance of downstream applications like automated candidate filtering.

Synonym and Variant Issues:

BERT sometimes struggles to recognize synonyms or closely related terms in domain-specific contexts. For instance, “cloud infrastructure” may not be equated with “AWS services,” despite their overlap in meaning. This can hinder accurate understanding in applications requiring domain-specific synonym resolution, such as job matching or content retrieval.

Cross-Section Matching Limitations:

BERT tends to unequally weigh different components of a text, such as soft skills, certifications, and technical skills. For instance, it may prioritize technical skills while underrepresenting certifications or soft skills like "leadership" or "communication." This imbalance can lead to incomplete or biased assessments in use cases like candidate profiling or performance evaluations.

**Gemini: Strengths and Weaknesses**

Gemini, designed primarily for text generation tasks, offers several advantages over models like BERT, particularly in contextual understanding and language fluency. However, it also comes with notable limitations:

Deeper Contextual Matching*:*

Gemini demonstrates an improvement in contextual understanding compared to BERT. However, it often prioritizes keyword matching over deeper semantic equivalence. For example, while it might match a job description (JD) mentioning “front-end development” with a resume listing “React.js,” it may still overlook nuanced connections between broader concepts like “UI/UX design” and “user interface optimization.”

Handling Long-Form Text:

Gemini's limited token capacity poses a significant challenge when processing long-form documents like resumes or JDs. Critical sections, such as certifications or detailed project descriptions, may be truncated, leading to incomplete analyses. This limitation is especially problematic in applications requiring comprehensive context preservation.

Poor Handling of Ambiguity:

Ambiguities in text, such as resumes listing multiple roles or hybrid skills, can confuse BERT. For example, a profile mentioning experience in both "software engineering" and "technical project management" may lead to inconsistent interpretations, impacting the relevance of downstream applications like automated candidate filtering.

Semantic Role Labeling (SRL):

While capable of identifying explicit roles and skills, Gemini often misses implied experiences or skills not directly stated in the text. For example, a JD implying leadership through phrases like “mentored junior developers” may not be fully captured, resulting in gaps in semantic role interpretation.

Domain-Specific Matching:

Gemini lacks inherent fine-tuning capabilities for specialized industries, limiting its effectiveness in niche domains such as healthcare, finance, or aerospace. Without targeted adaptations, it struggles to differentiate between domain-specific terms and general vocabulary, impacting the precision of applications like industry-specific job matching.

Synonym and Variant Issues:

Gemini relies heavily on lexical similarity, which limits its ability to identify nuanced equivalences between terms. For instance, it may not equate “machine learning pipelines” with “automated ML workflows,” resulting in missed matches. This limitation impacts scenarios where synonym recognition is critical for success.

**LLaMA : Advancements in Contextual Understanding**

LLaMA 3.1, a cutting-edge model for natural language processing, surpasses both BERT and Gemini in several key areas, demonstrating significant advancements in contextual understanding and semantic representation:

Superior Contextual Embedding:

LLaMA 3.1 excels in capturing nuanced relationships between various entities, such as skills, responsibilities, and experiences. For example, it can link “budget management” with “project leadership,” ensuring a more holistic understanding of resumes and JDs. This deeper embedding enables improved alignment between candidate profiles and job requirements.

Handling Long-Form Text:

Unlike Gemini, LLaMA 3.1 can process entire documents, such as resumes and job descriptions, in a single pass without truncation. This capability ensures that critical information—such as detailed project histories or advanced certifications—is retained, preserving section-level dependencies for accurate analysis

Semantic Role Labeling (SRL):

LLaMA 3.1 demonstrates robust semantic role labeling capabilities, accurately identifying relationships between roles, responsibilities, and skills. For example, it can infer leadership qualities from phrases like “managed cross-functional teams” or “oversaw project delivery,” even when such qualities are not explicitly stated.

Domain-Specific Matching:

LLaMA 3.1's open-source nature allows for fine-tuning to specific industries, such as technology, healthcare, or finance. This adaptability ensures high precision in understanding industry-specific terminologies, enabling tailored applications like specialized job matching or sector-specific content generation.

Synonym and Variant Issues:

The model demonstrates superior synonym and variant recognition, equating terms like “data analysis” with “data science” and “serverless architecture” with “cloud-native design.” This reduces false negatives and improves the relevance of results in applications requiring synonym resolution.

Ambiguity Handling:

LLaMA 3.1 is particularly adept at handling ambiguity, effectively disambiguating overlapping roles or hybrid skill sets. For example, in a resume listing “full-stack developer and technical writer,” it prioritizes the most relevant sections based on the context of a JD, ensuring accurate role alignment.

These advancements make LLaMA 3.1 a superior choice for tasks requiring deep contextual understanding, long-form document processing, and domain-specific adaptations, enabling more accurate and nuanced analyses in a wide range of applications.

**End-to-End Preprocessing Pipeline for Resume and Job Description Matching**

Resume Extraction from PDF

Since resumes are uploaded in PDF format, we use pypdf to extract raw text from the document.The extraction process ensures that no crucial information is lost while converting PDFs to text.

Text Cleaning & Normalization

A. Before analysis, the extracted text undergoes cleaning and normalization to remove inconsistencies.
B. Lowercasing: Converts all text to lowercase to avoid case sensitivity issues.
C. Whitespace Removal: Eliminates extra spaces, line breaks, and redundant formatting.
D. Punctuation Removal: Removes unnecessary symbols that do not contribute to meaning.
E. Stopword Removal: Common words (e.g., "the", "is", "and") that do not add value to the analysis are removed using nltk.

Tokenization (Splitting Text into Words & Sentences)

A. The cleaned text is split into individual words (word tokenization) to enable keyword-based processing.
B. Sentence tokenization is also performed to retain context in longer descriptions.
C. nltk.word_tokenize() and nltk.sent_tokenize() are used for this step**.**

Stemming & Lemmatization

A. Stemming reduces words to their root forms (e.g., "running" → "run"), improving text consistency.
B. Lemmatization ensures words are transformed into meaningful base forms while retaining grammatical accuracy (e.g., "better" → "good").
C. We use NLTK's Porter Stemmer and WordNet Lemmatizer for this step.

Named Entity Recognition (NER) for Key Information Extraction

Using AI-powered models like Gemini, we extract structured details from resumes:

A. Personal Details: Name, contact info, LinkedIn, GitHub, etc.
B. Education: Degree, institution, graduation year, and marks.
C. Work Experience: Company names, job titles, durations, and key responsibilities.
D. Skills & Certifications: Extracted and categorized based on job requirements.
E. Projects & Achievements: Identified for relevance to job descriptions.

Job Description Processing

A. Text is preprocessed in the same way as resumes (cleaning, tokenization, lemmatization).
B. Keyword extraction identifies the most relevant job requirements.
C. Skills and experience levels are detected using NLP techniques.

Vectorization & Feature Engineering

To make resumes and job descriptions machine-readable, we convert text into numerical vectors using:

A. TF-IDF (Term Frequency-Inverse Document Frequency) to highlight important terms.
B. Word Embeddings (HuggingFace BERT, Gemini AI) for deep semantic understanding.

C. Bag of Words (BoW) for frequency-based keyword matching.

Resume-Job Description Matching & Scoring
We evaluate resumes against job descriptions based on:
A. Keyword Matching: Ensuring essential skills are present.
B. Cosine Similarity: Measuring the closeness of resume and job description embeddings.
C. Weighted Scoring: Prioritizing key areas such as skills, experience, and education.
D. Missing Keywords & Suggestions: Identifying gaps in resumes and providing feedback for improvement.

**Architecture Diagram**

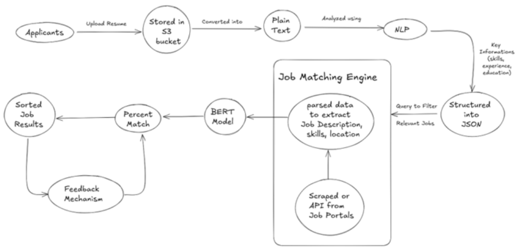


**Fig.1** Job Matching Engine Architecture

The Job Matching Engine is designed to analyze resumes and job descriptions to provide ranked job recommendations based on relevance. Applicants upload their resumes, which are stored in an S3 bucket and converted into plain text. NLP techniques extract key details such as skills, experience, and education, structuring them into a JSON format for further processing.

The system collects job listings through web scraping or API integration from various job portals. The extracted job descriptions are parsed to ensure a structured format, making it easier to match candidates with relevant roles. Using the parsed data, queries are formed to filter suitable job opportunities based on applicant information.

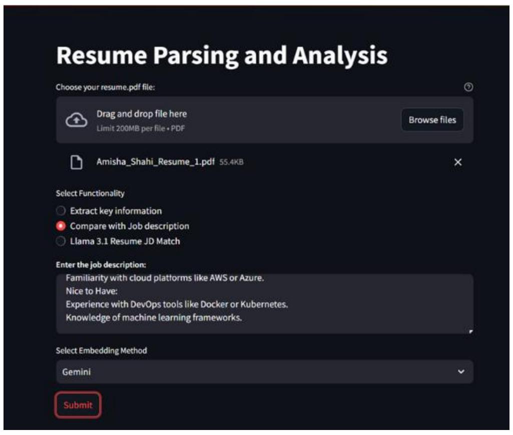
BERT model is used to compute the similarity between an applicant's resume and job descriptions. Instead of relying solely on keyword matching, the model processes semantic meaning, calculating a match percentage that determines how well an applicant's qualifications align with a given job.

The system then ranks the job results based on relevance and presents them to the user. A feedback mechanism allows applicants to interact with the recommendations, improving future matches by refining the model's understanding of preferences. This iterative approach ensures that over time, the job recommendations become more personalized and accurate.

## V. Results Visualization

The resume and the job description have been given as input, and we obtained three relevance scores based on Gemini, BERT, and LLaMA models. The following results depict their respective outputs:

**Job Matching Score Comparison Across Models**
**1.Gemini Result:** Gemini processes the input data, extracting key insights and matching relevant job criteria.



Resume Parsing and Analysis

Choose your resume.pdf file:

Drag and drop file here
Limit 200MB per file • PDF

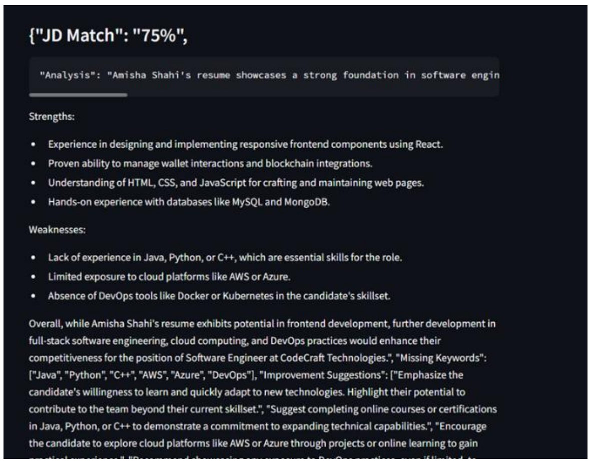
Amisha_Shahi_Resume_1.pdf 55.4KB

Select Functionality
Extract key information
[✓ Compare with Job description]
Llama 3.1 Resume JD Match

Enter the job description:
Familiarity with cloud platforms like AWS or Azure.
Nice to Have:
Experience with DevOps tools like Docker or Kubernetes.
Knowledge of machine learning frameworks.

Select Embedding Method
Gemini

**Fig.1** Analyzing Job Matching Using Gemini



{"JD Match": "75%",

```
"Analysis": "Amisha Shahi's resume showcases a strong foundation in software engin
```

Strengths:

- Experience in designing and implementing responsive frontend components using React.
- Proven ability to manage wallet interactions and blockchain integrations.
- Understanding of HTML, CSS, and JavaScript for crafting and maintaining web pages.
- Hands-on experience with databases like MySQL and MongoDB.

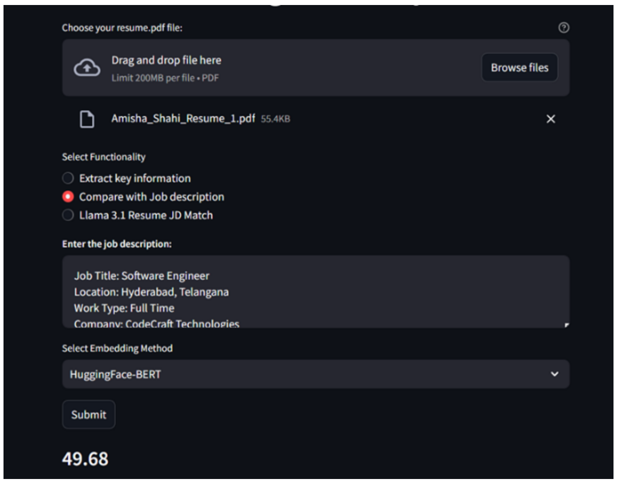
Weaknesses:

- Lack of experience in Java, Python, or C++, which are essential skills for the role.
- Limited exposure to cloud platforms like AWS or Azure.
- Absence of DevOps tools like Docker or Kubernetes in the candidate's skillset.

Overall, while Amisha Shahi's resume exhibits potential in frontend development, further development in full-stack software engineering, cloud computing, and DevOps practices would enhance their competitiveness for the position of Software Engineer at CodeCraft Technologies.", "Missing Keywords": ["Java", "Python", "C++", "AWS", "Azure", "DevOps"], "Improvement Suggestions": ["Emphasize the candidate's willingness to learn and quickly adapt to new technologies. Highlight their potential to contribute to the team beyond their current skillset.", "Suggest completing online courses or certifications in Java, Python, or C++ to demonstrate a commitment to expanding technical capabilities.", "Encourage the candidate to explore cloud platforms like AWS or Azure through projects or online learning to gain

**Fig. 2** Job Matching Score and Analysis Using Gemini

**2. BERT Score:** BERT computes a similarity score, measuring how well the resume aligns with the job description



Choose your resume.pdf file:

Amisha_Shahi_Resume_1.pdf 55.4KB

Select Functionality

( ) Extract key information
(●) Compare with Job description
( ) Llama 3.1 Resume JD Match

Enter the job description:

Job Title: Software Engineer
Location: Hyderabad, Telangana
Work Type: Full Time
Company: CodeCraft Technologies

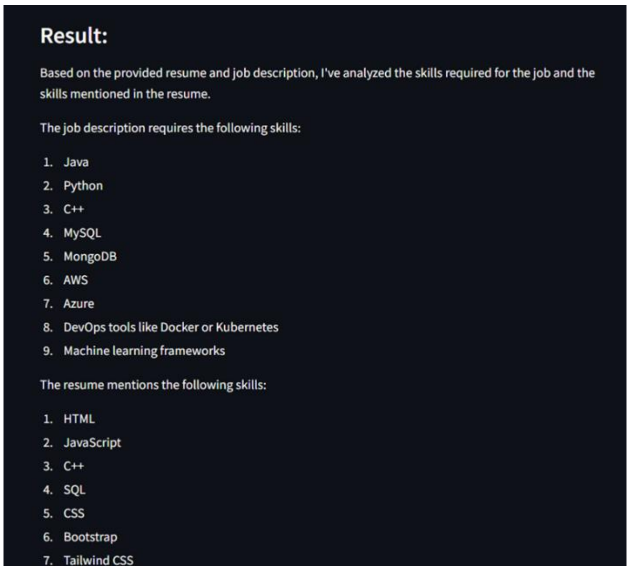
Select Embedding Method

HuggingFace-BERT

49.68

**Fig.3** Job Matching Score and Analysis Using BERT

**3. LLaMA Result:** LLaMA analyzes the contextual relevance between the resume and job description, ranking them accordingly.



Result:

Based on the provided resume and job description, I've analyzed the skills required for the job and the skills mentioned in the resume.

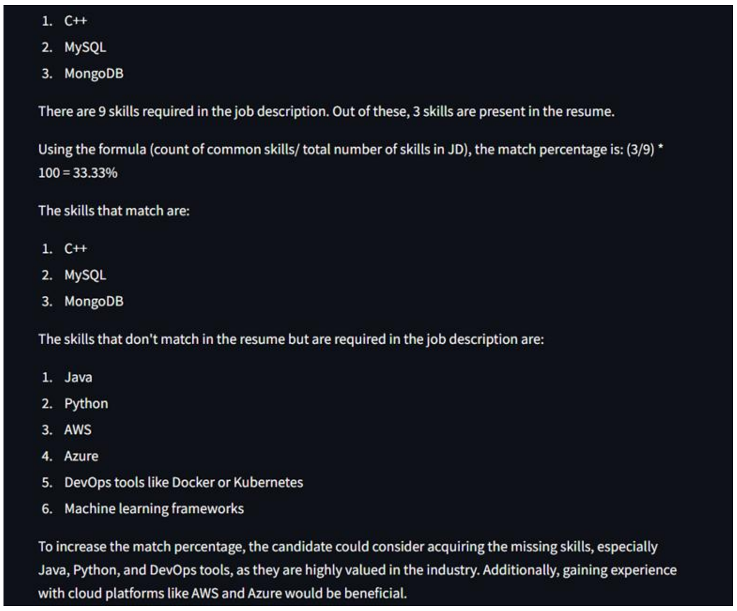
The job description requires the following skills:

1. Java
2. Python
3. C++
4. MySQL
5. MongoDB
6. AWS
7. Azure
8. DevOps tools like Docker or Kubernetes
9. Machine learning frameworks

The resume mentions the following skills:

1. HTML
2. JavaScript
3. C++
4. SQL
5. CSS
6. Bootstrap
7. Tailwind CSS

**Fig.4** Analyzing Job Matching Using Llama 3.1

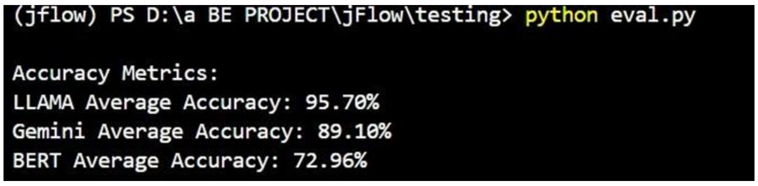


```
1. C++
2. MySQL
3. MongoDB

There are 9 skills required in the job description. Out of these, 3 skills are present in the resume.

Using the formula (count of common skills/ total number of skills in JD), the match percentage is: (3/9) *
100 = 33.33%

The skills that match are:

1. C++
2. MySQL
3. MongoDB

The skills that don't match in the resume but are required in the job description are:

1. Java
2. Python
3. AWS
4. Azure
5. DevOps tools like Docker or Kubernetes
6. Machine learning frameworks

To increase the match percentage, the candidate could consider acquiring the missing skills, especially
Java, Python, and DevOps tools, as they are highly valued in the industry. Additionally, gaining experience
with cloud platforms like AWS and Azure would be beneficial.
```

**Fig.5** Job Matching Score and Analysis Using Llama 3.1

**Model Accuracy Evaluation and Comparison**

The evaluation focused on the accuracy of each model across the dataset. The graphical representation of the results is provided in Figure 6, which visually compares the accuracy of the models.



```
(jflow) PS D:\a BE PROJECT\jFlow\testing> python eval.py

Accuracy Metrics:
LLAMA Average Accuracy: 95.70%
Gemini Average Accuracy: 89.10%
BERT Average Accuracy: 72.96%
```

**Fig. 6** Accuracy of each model across the dataset

LLAMA 3.1 model outperformed Gemini and BERT, achieving the highest average accuracy of 95.70%. The Gemini model showed competitive performance with an accuracy of 89.10%, while BERT lagged behind with an accuracy of 72.96%.

## VI. Discussion And Conclusions

Our study underscores the limitations of BERT and Gemini in resume-JD alignment tasks, particularly in contextual sensitivity and domain-specific adaptability. LLaMA 3.1 emerges as the superior model, offering advanced semantic understanding, enhanced synonym matching, and effective handling of long-form and ambiguous texts. These capabilities make LLaMA 3.1 the optimal choice for AI-driven recruitment platforms focused on precision and nuanced candidate matching

## References

[1] Devaraju, Sudheer. (2022). Natural Language Processing (NLP) In AI-Driven Recruitment Systems. International Journal Of Scientific Research In Computer Science Engineering And Information Technology. 8. 555-566. 10.32628/CSEIT2285241.

[2] Maitra, Mayukh & Sinha, Surabhi & Kierszenowicz, Tomas. (2024). An Improved BERT Model For Precise Job Title Classification Using Job Descriptions. 1-6. 10.1109/IS61756.2024.10705204.

[3] Prathima, V.. (2024). Resume Application Tracking System With Google Gemini Pro. International Journal For Research In Applied Science And Engineering Technology. 12. 66-70. 10.22214/Ijraset.2024.59645.

[4] Bhatt, A., Vaghela, N., & Dudhia, K. (2024). Generating Knowledge Graphs From Large Language Models: A Comparative Study Of GPT-4, Llama 2, And BERT. Https://Doi.Org/10.48550/Arxiv.2412.07412.

[5] Musleh, D. A. (2024). Rule-Based Information Extraction From Multi-Format Resumes For Automated Classification. Mathematical Modelling Of Engineering Problems, 11(4), 1044–1052. Https://Doi.Org/10.18280/Mmep.110422.

[6] Pudasaini, Shushanta & Shakya, Subarna & Lamichhane, Sagar & Adhikari, Sajjan & Tamang, Aakash & Adhikari, Sujan. (2022). Scoring Of Resume And Job Description Using Word2vec And Matching Them Using Gale–Shapley Algorithm. 10.1007/978-981-16-2126-0_55.

[7] Boddu, Sai Tarun & Desu, Sujeeth & Puli, Sreekanth. (2023). RESUME SUMMARIZER AND JOB DESCRIPTION MATCHER USING NATURAL LANGUAGE PROCESSING AND SPACY. INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. 07. 1-11. 10.55041/IJSREM26647.

[8] Devaraju, S. (2022). Natural Language Processing (NLP) In AI-Driven Recruitment Systems. International Journal Of Scientific Research In Computer Science, Engineering And Information Technology, 555–566. Https://Doi.Org/10.32628/Cseit2285241

[9] Lei, H., Lin, Y., Li, X., & Addo, P. (2016). Machine Learned Resume-Job Matching Solution. Https://Doi.Org/10.48550/Arxiv.1607.07657

[10] M.S. Rebrii, And S.L. Zinovatna, Candidates Resume Analyzer Using Artificial Intelligence, Informatics. Culture. Technology. 1 (2024) 139–146. Doi:10.15276/Ict.01.2024.20.

[11] M. Alghazal, Talent Acquisition Process Optimization Using Machine Learning In Resumes' Ranking And Matching To Job Descriptions, In: Society Of Plastics Engineers, 2021. Doi:10.2118/204534-Ms.

[12] L. Kotschenreuther, T. Zimmermann, And K. Schmidt, Data-Driven HR - Résumé Analysis Based On Natural Language Processing And Machine Learning, (2016). Doi:10.48550/Arxiv.1606.05611.

[13] S. Maheshwary, And H. Misra, Matching Resumes To Jobs Via Deep Siamese Network, In: Association For Computing Machinery, 2018: Pp. 87–88. Doi:10.1145/3184558.3186942.

[14] I. -, P. -, L. -, And M. -, NLP-Powered Resume Matching For Recruitment, International Journal For Multidisciplinary Research. 6 (2024). Doi:10.36948/Ijfmr.2024.V06i06.31742.

[15] C. Li, F. Er, R. Thomas, V. Hertzberg, S. Pittard, And J. Choi, Competence-Level Prediction And Resume & Job Description Matching Using Context-Aware Transformer Models, (2020). Doi:10.48550/Arxiv.2011.02998.

[16] R. Agarwal, A. Kulkarni, And V. James, Resume Shortlisting And Ranking With Transformers, In: Springer Nature Switzerland, 2023: Pp. 99–108. Doi:10.1007/978-3-031-35081-8_8.

[17] R. Shah, P. Rawat, A. Kumar, And V. Bhatia, End-To-End Resume Parsing And Finding Candidates For A Job Description Using BERT, (2019). Doi:10.48550/Arxiv.1910.03089.

[18] X. Li, H. Shu, Z. Lin, And Y. Zhai, A Method For Resume Information Extraction Using BERT-Bilstm-CRF, In: Institute Of Electrical Electronics Engineers, 2021: Pp. 1437–1442. Doi:10.1109/Icct52962.2021.9657937.

[19] R. Lakatos, P. Pollner, A. Hajdu, And T. Joó, Investigating The Performance Of Retrieval-Augmented Generation And Domain-Specific Fine-Tuning For The Development Of AI-Driven Knowledge-Based Systems, Machine Learning And Knowledge Extraction. 7 (2025) 15. Doi:10.3390/Make7010015.