ResumeMatch: A Dual-Role AI-Driven Platform for Transparent Resume-Job Compatibility and Enhanced Hiring Outcomes

by
Rabia Danish

A Major Research Project
presented to Toronto Metropolitan University
in partial fulfillment of the requirements for the degree of

Master of Science
in the Program of
Data Science and Analytics

Toronto, Ontario, Canada 2025

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my supervisor, Professor Dr. Pawel Pralat, for his invaluable guidance and unwavering support throughout this project. His expertise and insightful feedback were crucial to the successful completion of this project.

ABSTRACT

Traditional Applicant Tracking Systems (ATS) and keyword-based recruitment models often fail to capture the contextual and semantic nuances of professional experience, leading to sub-optimal candidate–job alignments. This reliance on lexical matching can penalize qualified candidates due to formatting inconsistencies or missing keywords, while hindering recruiters' ability to recognize transferable skills and candidate potential. This research proposes ResumeMatch, a dual-role AI-driven platform that enhances resume–job alignment through structured information extraction and semantic similarity modeling.

Three stages of experimentation were conducted: (1) Job Description Parsing, (2) Resume Parsing, and (3) Resume–Job Matching. For entity recognition, a custom transformer-based Named Entity Recognition (NER) model was trained on 150 manually annotated job descriptions to extract hard skills, soft skills, education, and experience. Evaluation was conducted on a 100-JD hold-out set with LLM-assisted labels, benchmarking ten extraction methods covering rule-based, taxonomy-driven, transformer-based, and LLM-based approaches. Results demonstrate that the custom NER achieved the strongest overall performance for skills extraction, while the LLM-based method performed best for education and recall on experience.

For resume–job alignment, four similarity models (TF-IDF, LDA, Word2Vec, and BERT) were compared using unsupervised metrics. LDA achieved the best ranking separation, while BERT embeddings delivered the highest reciprocal agreement, lowest mutual rank, and strongest cluster separation, highlighting complementary strengths across methods.

A prototype system was implemented in Streamlit, enabling weighted similarity scoring, clustering-based candidate ranking, skill-gap analysis, and interview question generation. The framework demonstrates a scalable, explainable, and data-driven approach to improving recruitment efficiency and candidate fairness.

Keywords: ResumeMatch, NLP, Resume parsing, Job matching, Skill extraction, Named Entity Recognition, BERT, LLMs, Semantic similarity, Clustering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background

Artificial Intelligence (AI) is transforming the landscape of talent acquisition, streamlining the processes of resume screening, job matching, and applicant evaluation. Recruitment tools now incorporate machine learning (ML), natural language processing (NLP), and deep learning algorithms to provide scalable solutions that can parse unstructured data and identify promising candidates. Despite these advancements, challenges remain in ensuring fairness, transparency, interpretability, and real-world applicability.

Conventional Applicant Tracking Systems (ATS) often use rigid, keyword-based filters that disproportionately penalize candidates for stylistic differences or missing keywords. They fail to recognize semantically similar expressions or transferable skills, resulting in missed hiring opportunities. On the other hand, recruiters are overwhelmed by large volumes of applications and require intelligent tools that go beyond surface-level filtering.

ResumeMatch was conceived to address these issues. It is a dual-role AI platform that leverages modern NLP models and LLMs to provide transparency and contextual understanding in resume-job compatibility. Job seekers receive personalized, actionable feedback, while recruiters benefit from data-driven candidate rankings and interactive dashboards.

This paper details the development process of ResumeMatch, starting with a comprehensive literature review and an extensive exploratory data analysis. The final platform is intended to support ethical, efficient, and equitable hiring decisions.

1.2 Research Problem

The primary research problem addressed by this project stems from the inefficiencies and inherent biases in traditional and existing automated recruitment systems. Current Applicant Tracking Systems (ATS) often rely on simplistic keyword matching, leading to the rejection of qualified candidates whose resumes do not perfectly align with predefined terms, even if they possess relevant transferable skills or experience. This results in a significant volume of false negatives for job seekers and a substantial manual screening burden for recruiters. Furthermore, the lack of transparency in many AI-driven hiring tools exacerbates concerns about fairness and interpretability, making it difficult for users to understand why certain decisions are made.

1.3 Objectives

Given the problem, the objectives of this research project are as follows:

1. Develop and evaluate robust parsing models: To create and assess NLP models capable of extracting structured information (hard skills, soft skills, education, and experience) from unstructured job descriptions and resumes.

2. Design and implement a semantic matching engine: To compute semantic similarity between parsed job descriptions and resumes using advanced embedding techniques, thereby enabling nuanced and accurate candidate–job compatibility assessments.
3. Enhance transparency and explainability: To integrate interpretable features, such as visualizations of matched and missing skills, enabling both job seekers and recruiters to gain actionable insights.
4. Reduce recruitment latency: To automate resume clustering and job–description matching, significantly decreasing the time required for manual screening.

1.4 Data Sources and Scope

The system is developed using two comprehensive datasets sourced from Kaggle:

1. **LinkedIn Job Postings Dataset**: Contains a comprehensive record of 123,849 job postings from 2023–2024, comprising title, description, salary, location, company metadata, work type, and skills.
2. **Resume Dataset**: Contains over 2,400 resumes, each labeled by job role category and available in both text and PDF formats.

The scope includes text cleaning, salary normalization, skill extraction, and embedding generation. These components are used to build a semantic similarity-based system supported by deep exploratory analysis.

1.5 Contributions

This project makes the following contributions:

- **Custom NER Development:** Trained a transformer-based spaCy NER model on 150 Doccano-annotated job descriptions to extract hard skills, soft skills, education, and experience.
- **Evaluation Framework:** Introduced a 100-job description hold-out set, annotated with LLM assistance, and benchmarked ten extraction methods across multiple metrics (Precision, Recall, F1, Jaccard).
- **Unsupervised Matching Evaluation:** Compared four embedding approaches (TF-IDF, LDA, Word2Vec, BERT) using unsupervised metrics including Separation@K, Reciprocal Top K Agreement, Mutual Rank, and Cluster Agreement.
- **Prototype System:** Built a Streamlit-based dual-role interface for job seekers and recruiters, featuring weighted similarity scoring, clustering-based candidate triage, skill-gap analysis, and interview question generation.

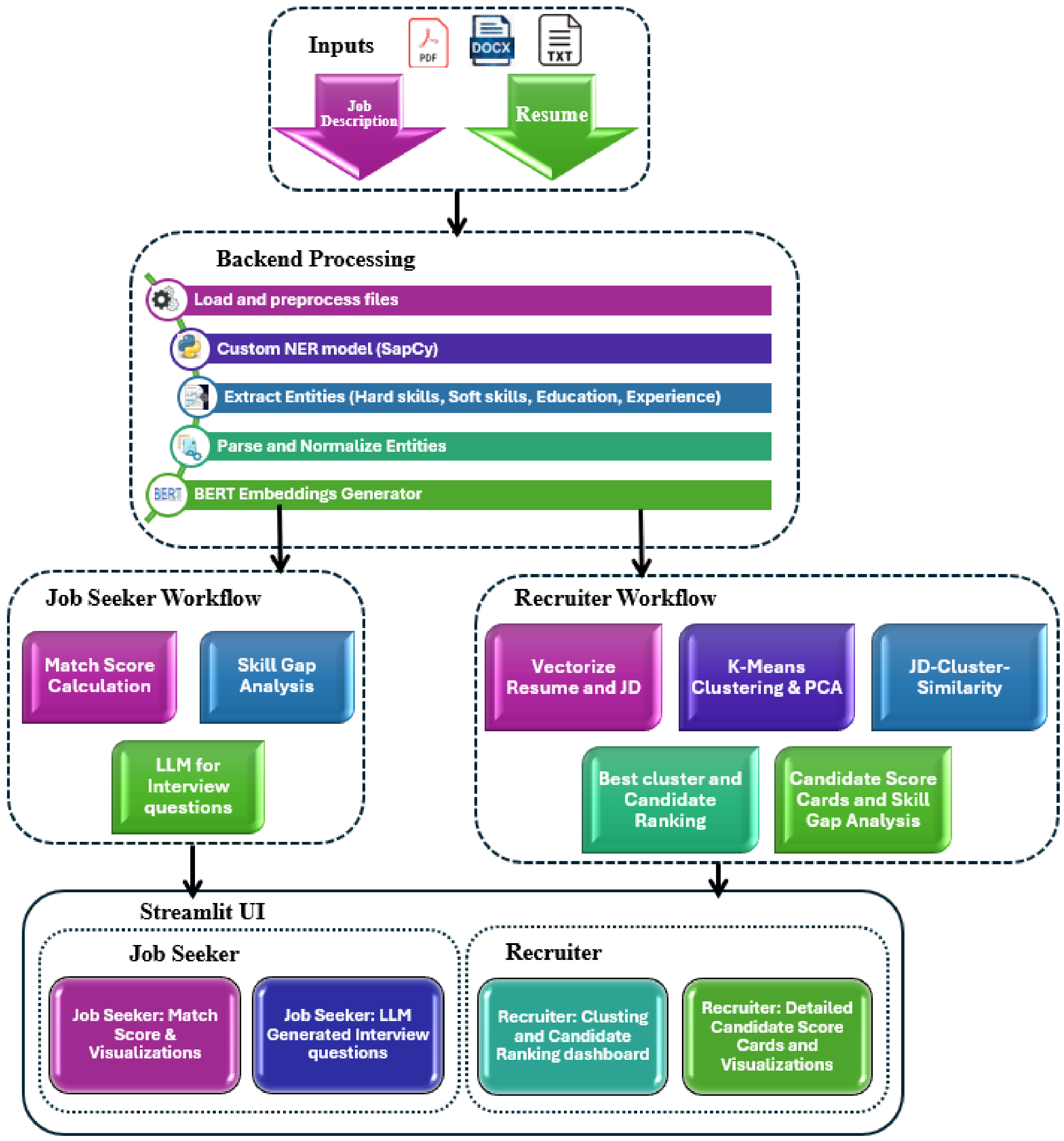

Inputs
PDF
DOCX
TXT
Job Description
Resume
Backend Processing
Load and preprocess files
Custom NER model (SapCy)
Extract Entities (Hard skills, Soft skills, Education, Experience)
Parse and Normalize Entities
BERT Embeddings Generator
Job Seeker Workflow
Match Score Calculation
Skill Gap Analysis
LLM for Interview questions
Recruiter Workflow
Vectorize Resume and JD
K-Means Clustering & PCA
JD-Cluster-Similarity
Best cluster and Candidate Ranking
Candidate Score Cards and Skill Gap Analysis
Streamlit UI
Job Seeker
Job Seeker: Match Score & Visualizations
Job Seeker: LLM Generated Interview questions
Recruiter
Recruiter: Clusting and Candidate Ranking dashboard
Recruiter: Detailed Candidate Score Cards and Visualizations

Figure 1.1: ResumeMatch Architecture Overview

2. LITERATURE REVIEW

The adoption of artificial intelligence (AI) in talent acquisition has been a major research focus, particularly in the areas of resume analysis, job recommendation systems, and algorithmic fairness. This review synthesizes key findings from a wide range of peer-reviewed studies to provide a comprehensive understanding of the current landscape, methodologies, and persistent challenges.

A significant area of research is dedicated to advanced resume analysis and information extraction. Historically, this task relied on rule-based methods, but recent studies have demonstrated the superior performance of deep learning architectures. For example, Ayishathahira et al. (2018) successfully employed a combination of Convolutional Neural Networks (CNNs) and Bi-LSTM-CRF models to parse 23 distinct fields from resumes, highlighting the superiority of deep learning over traditional methods. More recently, Kinger et al. (2024) integrated YOLOv5 for section identification and DistilBERT for named entity recognition, achieving a parsing accuracy of 96.2%. However, these models often face challenges with domain specificity, language diversity, and different resume formats. In an effort to address scalability, Bhatt et al. (2024) utilized MapReduce techniques to efficiently preprocess large-scale resume datasets, which led to a 97.2% F1-score in their downstream machine learning algorithms. This demonstrates the potential of combining big data processing with classification algorithms, though the authors noted a lack of standardized benchmarks and varied performance across diverse resume structures and industries.

The field has also seen a significant shift toward developing sophisticated recommender systems that use embeddings for semantic matching. Bevara et al. (2025) introduced Resume2Vec, a system that leverages transformer-generated embeddings to align resumes with job descriptions more accurately than traditional keyword-based applicant tracking systems (ATS). Their work showed marked improvements in ranking accuracy using Normalized Discounted Cumulative Gain (nDCG) and Ranked Biased Overlap (RBO), particularly in mechanical engineering and healthcare. However, the use of student raters instead of HR professionals for human evaluation weakened the study's external validity, and challenges with interpretability and computational efficiency remain unresolved for production-level deployment. For large-scale applications, Zhao et al. (2021) implemented a two-stage embedding-based system that combined fused embeddings and contextual reranking, which resulted in a 104% increase in click-through rate and a 37% improvement in nDCG over baselines. This validated the use of deep representation learning in production environments, though concerns persist around computational load and reproducibility due to proprietary data and algorithms. Similarly, the CCRS model by Özcan & Ögüdücü (2017) addressed the cold start problem and data sparsity by creating a reciprocal matching model that accounts for both candidate and employer preferences. By incorporating TrustRank and user interaction features, CCRS outperformed baseline classifiers, yet its limited dataset (7455 applications) and lack of demographic diversity raise questions about the model's robustness in varied recruitment scenarios.

Ethical considerations, particularly fairness and bias, have become a recurring theme in recent research. Delecraz et al. (2022) developed a job-matching algorithm specifically for NEET (Not in Employment, Education, or Training) populations and incorporated specific fairness metrics like Disparate Impact and Statistical Parity. Their study uncovered systemic underrepresentation in job recommendations, particularly for foreign nationals and applicants requiring work permits, emphasizing that algorithmic fairness requires deeper structural changes in data collection and organizational policy. Frissen et al. (2023) proposed a system to classify discriminatory language in job advertisements into five categories using Random Forest classifiers and FastText embeddings. While effective in identifying exclusionary language, its scope was limited to job descriptions, and the authors recommended integrating such systems into the hiring pipeline to promote holistic diversity and inclusivity, including the creation of fairness scores. Furthermore, user perception of these tools is critical, as Zhang and Yencha (2022) found widespread skepticism among users regarding the fairness and effectiveness of AI-driven tools, particularly video interview screening tools. Resume-screening algorithms were marginally more acceptable, but fairness and effectiveness were questioned across demographic groups, with higher income and education levels correlating with more favorable perceptions. This underscores the need for user-centered design alongside algorithmic improvements. Burrell and Mcandrew (2023) underscored the need for continuous audits, inclusive datasets, and cultural change to combat discrimination in sensitive sectors like healthcare. Their consultant-based intervention highlighted that AI systems must be embedded within an organizational culture that prioritizes ethical hiring, calling attention to policy development and systemic accountability rather than just technical solutions.

Other notable research includes named entity recognition (NER) for specific resume components. Gaur et al. (2021) developed a semi-supervised deep learning approach to identify educational institutions and degrees with minimal labeled data, achieving 92.06% accuracy. This technique addressed the challenge of scarce annotated resume datasets, though it remained constrained to the education section, and the absence of a standard dataset for NER tasks in resumes continues to hinder benchmarking. Research also investigates forecasting labor market trends; Fettach et al. (2025) used temporal knowledge graph embeddings to predict skill demand in the Moroccan IT sector, demonstrating the feasibility of time-aware link prediction for anticipating emerging job skill requirements. Limitations included the lack of integration with real-time labor data and difficulty applying the model in open-world settings. A few studies combined AI systems with real-time user interaction features. Singh and Gupta (2023) designed a recommendation engine that incorporated feedback loops, skill ontologies, and social boosting mechanisms to dynamically recalibrate candidate rankings. This improved screening speed and precision, but authors noted practical challenges such as organizational reluctance to invest in AI and the need for more transparent algorithmic decision-making. Building on real-time systems, J et al. (2024) proposed a CNN-based resume analyzer that predicted suitable job roles with 99.48% accuracy and generated feedback and course suggestions based on resume-job description compatibility. While outperforming traditional ML algorithms, limitations like restricted language support and oversimplified resume scoring metrics point to gaps in inclusivity and nuance. The authors recommended increasing dataset diversity and adopting more sophisticated pattern-matching techniques, reinforcing the need for richer, more representative training data. Other systems also aim to match resumes with jobs while providing applicant feedback. Pabalkar et al. (2024) presented an automated screening system combining TF-IDF and nine classifiers to match resumes to job descriptions, providing scoring and recommendations. Their Linear SVM model achieved 78.53% accuracy but struggled with precision compared to deep learning counterparts. The authors advocated for integrating with social media profiles (e.g., GitHub, LinkedIn) to improve context and personalization, aligning with growing interest in multi-modal recruitment systems.

2.1 Limitations and Future Directions

While recent studies demonstrate progress in AI-driven recruitment, several persistent limitations remain. A major concern is the lack of standardized, large-scale, publicly available datasets. Many investigations rely on small or proprietary datasets, often fewer than 15,000 resumes, which limits generalizability and hampers reproducibility across studies (Gaur et al., 2021). This reliance constrains the applicability of trained models in real-world settings and complicates consistent benchmarking.

Another recurring issue is the gap between algorithmic predictions and actual hiring outcomes. Although many models report high accuracy or ranking performance, few are validated using longitudinal data or real hiring decisions, which weakens confidence in their reliability. Zhao et al. (2021) addressed this by integrating user interaction metrics, yet comprehensive industry validation remains scarce.

Most recruitment systems also remain restricted in their linguistic and cultural scope. The majority are trained exclusively on English-language data, limiting their applicability to multilingual or global contexts. Fairness interventions, when attempted, often focus on individual demographic variables such as gender or nationality without addressing intersectionality or broader socio-economic influences (Bevara et al., 2025; Delecraz et al., 2022).

Explainability is another persistent challenge. Many AI models operate as opaque black boxes, offering limited transparency into recommendation processes. This lack of interpretability undermines user trust in recruitment contexts where accountability is essential. While some studies propose visualization or interpretability mechanisms, user-centered design practices are seldom applied.

Ethical and legal considerations are similarly underexplored. Few models directly address compliance with frameworks such as GDPR or EEOC guidelines. Scholars such as Burrell and McAndrew (2023) and Frissen et al. (2023) emphasize the importance of fairness audits and ongoing evaluations, yet systematic integration of safeguards into recruitment systems remains limited.

Finally, static models fail to adapt to evolving labor market demands. Many tools cannot incorporate dynamic learning or real-time data, reducing their long-term utility. Fettach et al. (2025) introduced temporal embeddings for skill forecasting, illustrating the potential of adaptive approaches. Building on this idea, the current research incorporates real-time job market data and dynamic skill recommendation modules supported by LLMs.

2.2 Motivation for ResumeMatch

The motivation for developing ResumeMatch stems from critical shortcomings observed in the contemporary recruitment landscape, particularly concerning the efficiency and fairness of candidate-job matching processes. Both job seekers and recruiters face significant pain points that current technologies often fail to adequately address.

For job seekers, the experience of applying for jobs can be frustrating and opaque. Many highly qualified individuals are overlooked due to rigid keyword filtering by Applicant Tracking Systems (ATS), which struggle to recognize transferable skills, synonyms, or varied phrasing of qualifications. This leads to a high volume of "false negatives," where suitable candidates are prematurely discarded. Furthermore, job seekers often lack clear, actionable feedback on why their applications were unsuccessful or how to improve their resumes to better align with job requirements. This opacity can be disheartening and hinder their career progression.

From the recruiter's perspective, the sheer volume of applications for a single role can be overwhelming. Manually sifting through hundreds or thousands of resumes is time-consuming and prone to human error and unconscious biases. While existing AI tools aim to streamline this process, many operate as "black boxes," providing rankings without transparent explanations. This lack of interpretability can erode trust in the system's recommendations and make it difficult for recruiters to justify hiring decisions or identify the true potential of diverse candidates. The challenge of identifying nuanced skills, particularly soft skills and those implicitly described in experience sections, further complicates effective candidate evaluation.

ResumeMatch is specifically designed to address these multifaceted challenges. By leveraging advanced NLP and large language models (LLMs), it moves beyond superficial keyword matching to capture the semantic meaning of skills, experience, and education. This deeper understanding improves compatibility assessments, reduces false negatives for job seekers, and streamlines the screening process for recruiters. The system uniquely supports dual user flows: a recruiter dashboard and job seeker reports, features that prior approaches often considered separately. In addition, ResumeMatch emphasizes transparency through skill gap visualization and detailed compatibility breakdowns, directly tackling the issue of interpretability. By delivering actionable insights and personalized feedback, the platform fosters a more efficient, equitable, and user-centric recruitment ecosystem.

3. DATA PREPARATION AND EXPLORATORY DATA ANALYSIS

This chapter details the crucial steps involved in preparing and understanding the datasets used for the ResumeMatch project. It covers data acquisition, cleaning, preprocessing, feature engineering, and a comprehensive exploratory data analysis (EDA) to uncover patterns and relationships within the data.

3.1 Data Acquisition

This research leveraged two publicly available datasets sourced from Kaggle: (1) the “linkedin-job-postings” dataset, containing detailed job advertisement data, and (2) the “resume-dataset,” consisting of categorized resume samples. The job dataset included multiple interrelated CSV files such as postings.csv, job_skills.csv, salaries.csv, and companies.csv while the resume dataset was composed of a single Resume.csv file and resume PDF files categorized by domain. Both datasets were loaded into pandas DataFrames via the kagglehub library.

3.2 Data Cleaning and Preprocessing

A comprehensive data cleaning pipeline was implemented to standardize, merge, and refine both datasets. The objective was to consolidate job posting details, company information, salary ranges, and associated skills into a single, cohesive dataset (job postings), and to prepare the resume data (resume_df).

1. Skill Aggregation: Job-level skills from job_skills.csv were joined with master skill definitions from skills.csv. These skills were then aggregated by job_id into a comma-separated list, forming an aggregated list of required skills per job posting.

2. Company Information Consolidation: Data from companies.csv and company_industries.csv were merged. Company industries were also aggregated by company_id into a list of industries associated with each company.

3. Salary Normalization: The salaries.csv table included minimum, maximum, and median salary figures across multiple pay periods (hourly, weekly, biweekly, monthly, and yearly). To ensure consistency, these raw values were converted into a standardized estimated annual salary using predefined conversion heuristics. For instance, hourly wages were converted by assuming full-time employment (40 hours/week × 52 weeks/year). These conversions assume no unpaid leave and may overestimate compensation in cases of part-time or contract roles. The normalized annual salary was subsequently merged into the main dataset for consistent analysis.

4. Sequential Merging for job_postings: Sequential merging used postings.csv as the primary table. Annualized salary, company metadata, industry lists, and aggregated skills were incrementally merged. Columns with excessive missing values, such as skills_description (over 121,410 missing), redundant salary columns and other non-meaningful columns were dropped.

5. Duplicate Removal: Duplicates were removed based on job_id for job postings (no duplicates found) and Resume_str for resumes (2 duplicates removed).

6. Handling Missing Values and Standardization: Missing values in remote_allowed were filled with 0, and the column was converted to a boolean type. The salary column was cast to a numeric data type. Text fields (e.g., company_name, pay_period, job_posting_location, etc.) were standardized to lowercase and stripped of leading/trailing whitespace. Missing values were imputed with "Not Specified." Rows missing job descriptions were removed. The resume dataset did not contain any missing values.

7. Text Cleaning: Special characters and formatting issues were removed from job description, and resume text using a custom preprocessing function. This included ASCII normalization, newline and whitespace handling, punctuation spacing, heuristic token separation (e.g., camelCase to camel Case), and bullet normalization. Optionally, lemmatization using spaCy was applied. The cleaned text fields were saved as new columns.

3.3 Feature Engineering

After standardization, additional features were engineered to enhance the dataset for downstream tasks. The binary field remote_allowed was cast to a boolean type. Outliers in the normalized annual salary were identified using the Interquartile Range (IQR) method. Specifically, any salary beyond 1.5 × IQR above the third quartile or below the first quartile was removed to reduce skewness and improve robustness in salary-related analyses. NaN values (representing missing salaries) were preserved. The job_posting_location field was parsed to extract city and state components from the raw string. While domain-level skill categories (e.g., marketing, sales) were included, they were retained primarily for supporting the later construction of a structured skill database (SKILLS_DB), as they lacked the necessary granularity for immediate use in modeling. The final cleaned and merged dataset was used for all subsequent analysis and model development steps.

3.4 Dataset Overview

After the comprehensive cleaning and preprocessing steps, the datasets were prepared for analysis. The job postings dataset, initially comprising over 123,849 records, was refined to approximately 122,890 unique entries after removing duplicates and handling outliers. This DataFrame contained 15 columns, providing detailed information such as job title, company name, location, salary, work type, experience level, skills, and industry. The resume dataset, consisted of 2,482 resumes, each with an ID, Category, and cleaned resume text.

3.5 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was crucial for uncovering patterns, anomalies, and relationships within the data. Python, with its rich ecosystem of data science libraries including NumPy for numerical operations, Pandas for data manipulation, Seaborn and Matplotlib for static visualizations, and Plotly for interactive plots, was instrumental for this purpose.

3.5.1 Univariate Analysis (Job Postings)

This section presents the distribution of individual variables within the job postings dataset. The histograms for job description lengths (Figure 3.1) were both right-skewed, indicating that while most job descriptions were relatively concise, a smaller number of significantly longer descriptions drove the mean length to be greater than the median.

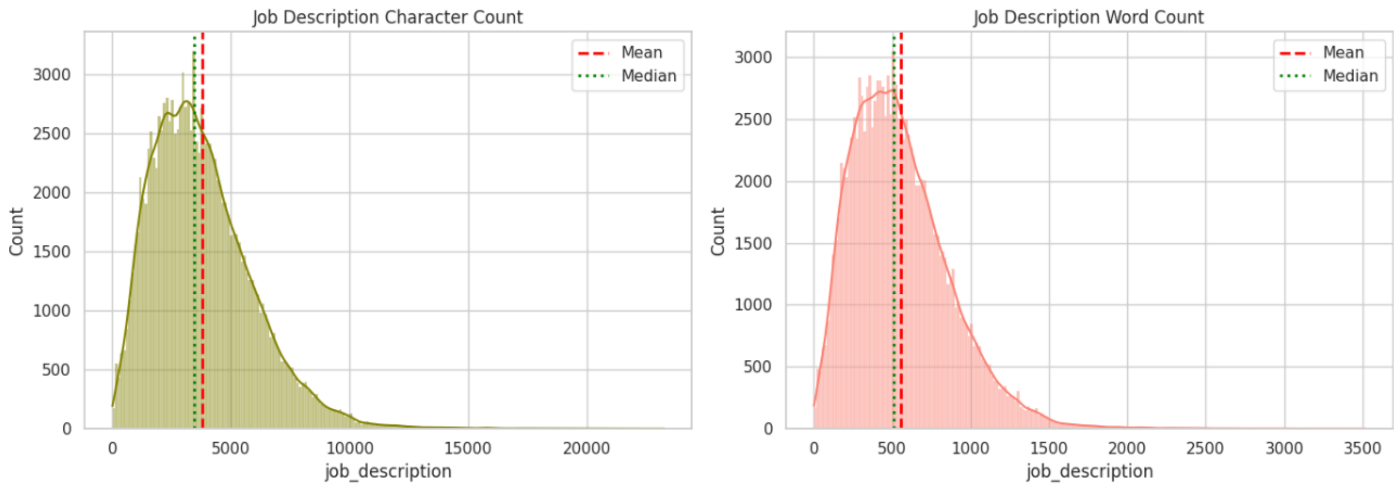


Figure 3.1: Job Description Character and Word Count Distribution

The distribution of final yearly salary (Figure 3.2) was also heavily right-skewed, with the mean salary ($205,328) significantly higher than the median ($81,500) due to high-end outliers. Most postings offered salaries between $52,000 and $125,000.

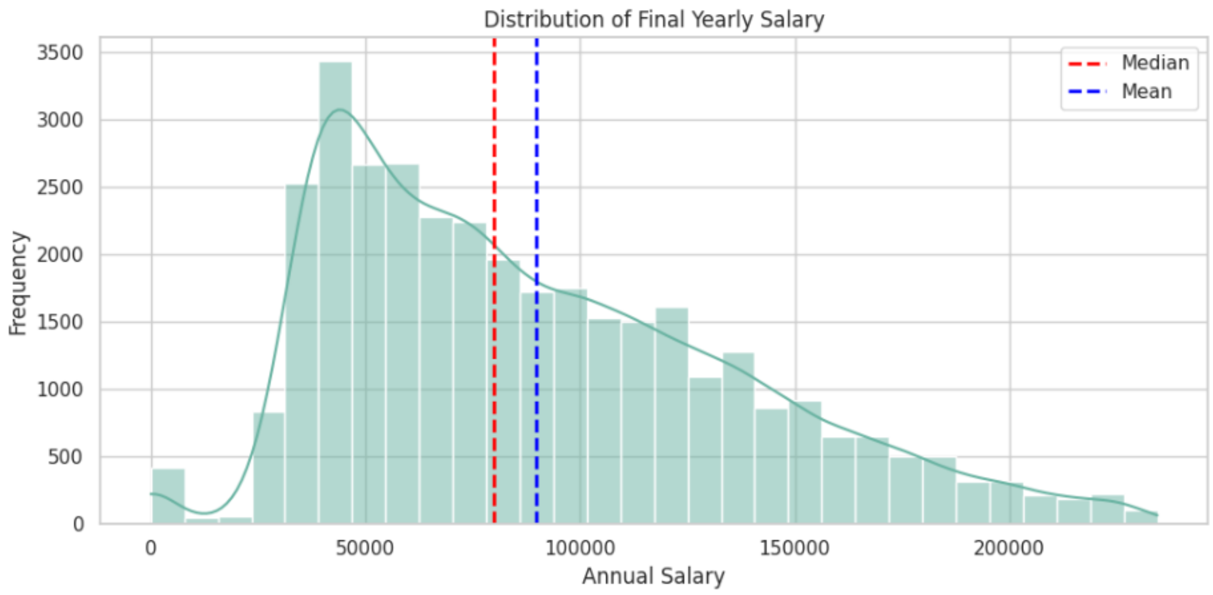


Figure 3.2: Distribution of Final Yearly Salary

An analysis of categorical variables highlighted key trends. The Work Type distribution (Figure 3.3) was dominated by 'full-time' positions, with 'contract' and 'part-time' roles appearing far less frequently. Similarly, the Experience Level distribution (Figure 3.4) showed that 'mid-senior level' and 'entry level' were the most common, while 'director' and 'executive' roles were less prevalent. Plots for Currency (Figure 3.5) and Pay Period (Figure 3.6) confirmed that 'USD' was the primary currency and 'yearly' and 'hourly' were the most common pay periods.

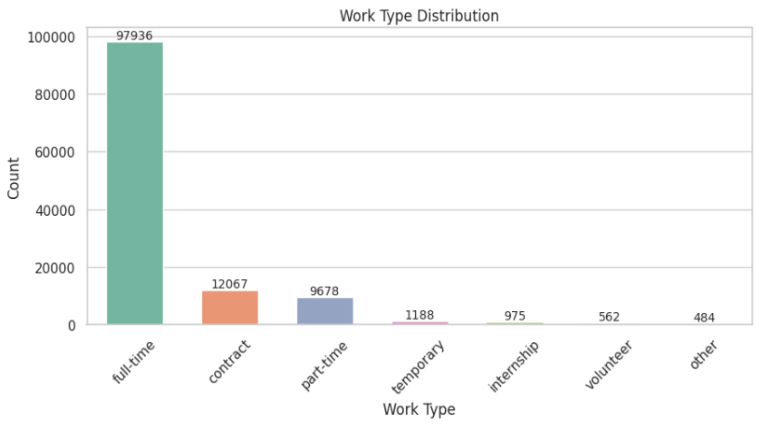


Figure 3.3: Work Type Distribution

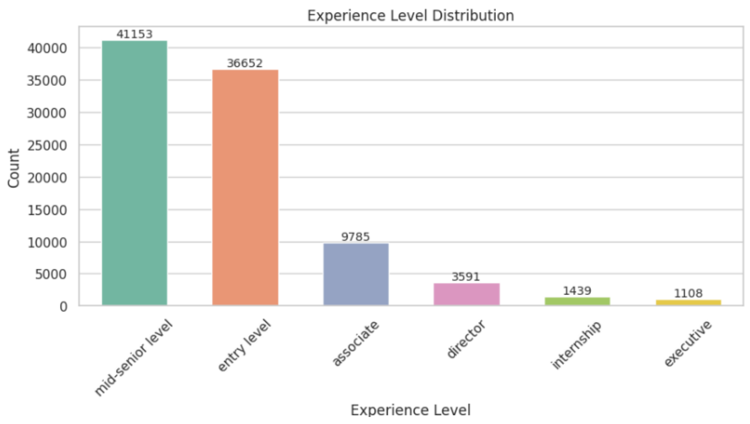


Figure 3.4: Experience Level Distribution

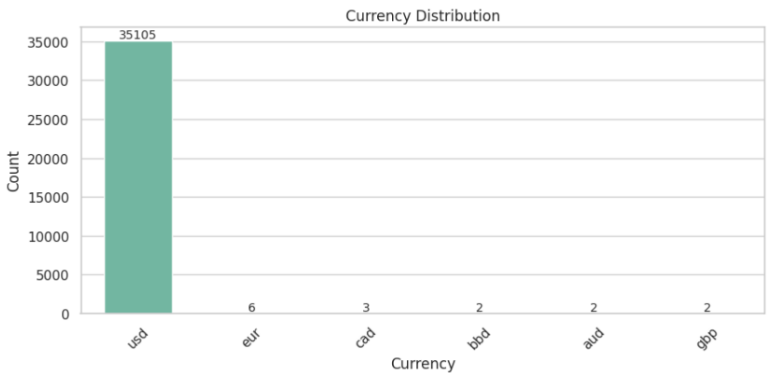


Figure 3.5: Currency Distribution

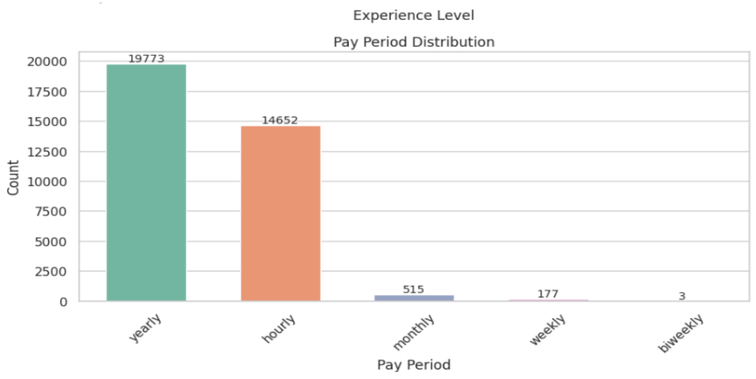


Figure 3.6: Pay Period Distribution

Finally, geographical and company-specific distributions identified the most active entities in the dataset. "Liberty healthcare and rehabilitation services" and "the job network" were the top companies by job postings (Figure 3.7). The top locations were led by states like California, Texas, and New York (Figure 3.8), with their major cities such as New York, Chicago, and Houston representing the top cities (Figure 3.9). The Remote Work Availability pie plot (Figure 3.10) revealed that only 12.2% of jobs allowed remote work, indicating that most roles were on-site or hybrid.

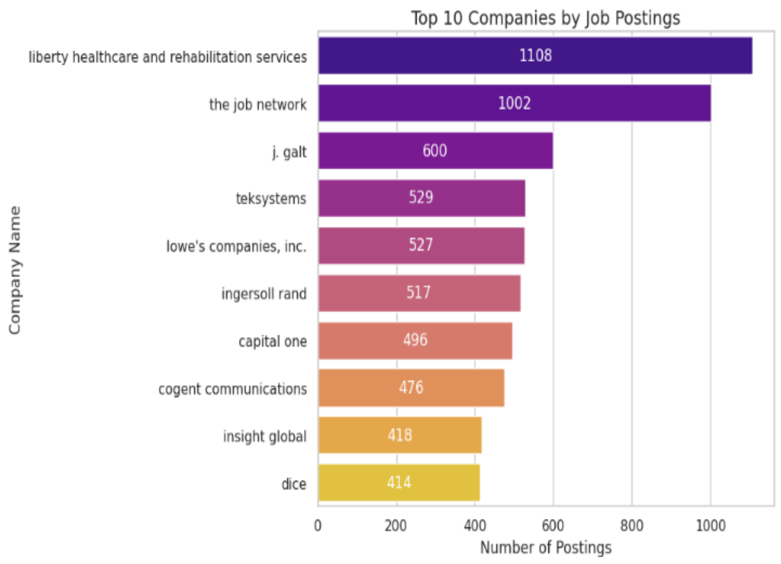


Figure 3.7: Top 10 Companies by Job Postings

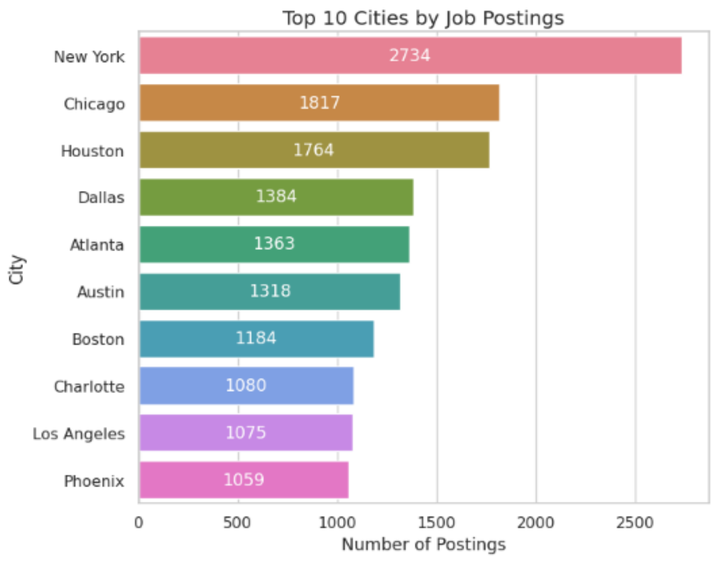


Figure 3.8: Top 10 Cities by Job Postings

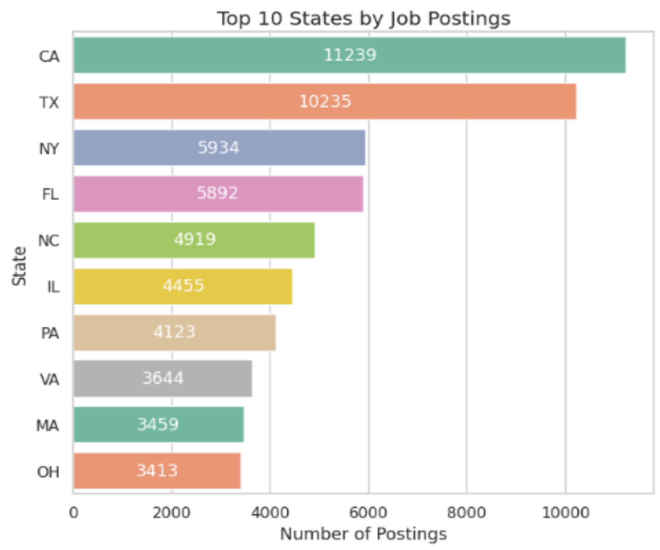


Figure 3.9: Top 10 States by Job Postings

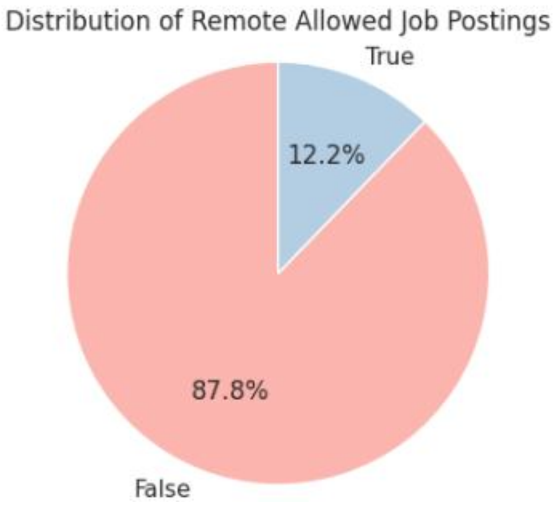


Figure 3.10: Remote Work Availability

3.5.2 Univariate Analysis (Resumes)

Univariate analysis was also applied to the resume dataset to understand its characteristics. Similar to job postings, the resume word count and character count distribution (Figure 3.11) was right-skewed, suggesting a majority of resumes were of moderate length, with a long tail of significantly longer documents.

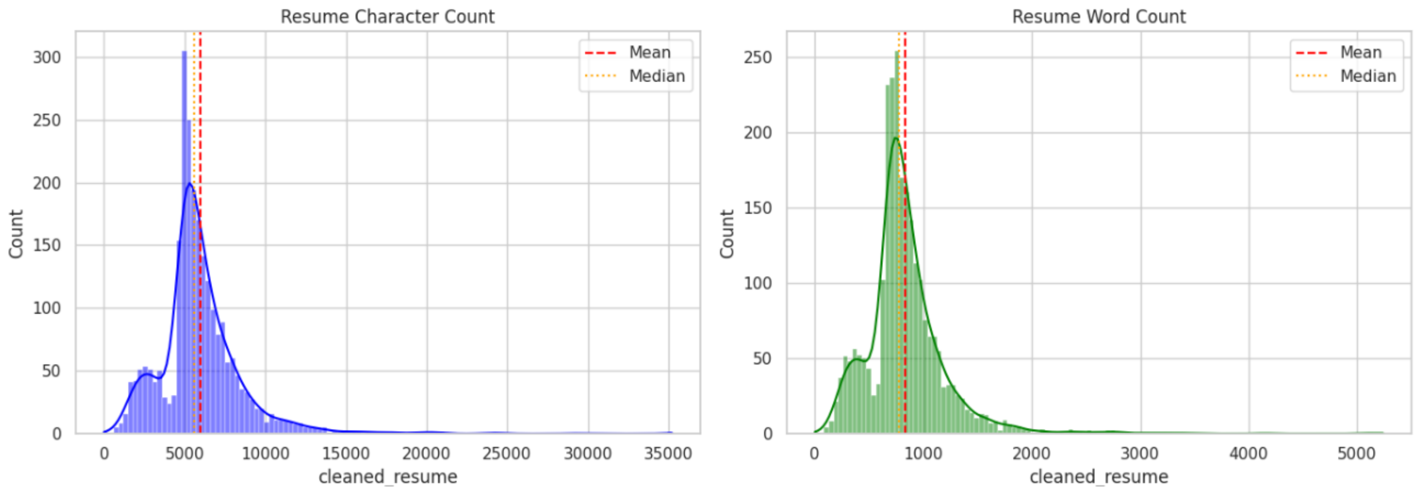


Figure 3.11: Resume Character and Word Count Distribution

The distribution of resume categories (Figure 3.12) revealed the frequency of resumes across different professional fields, with Information Technology, Business Development, Engineering, and Accounting having the highest counts.

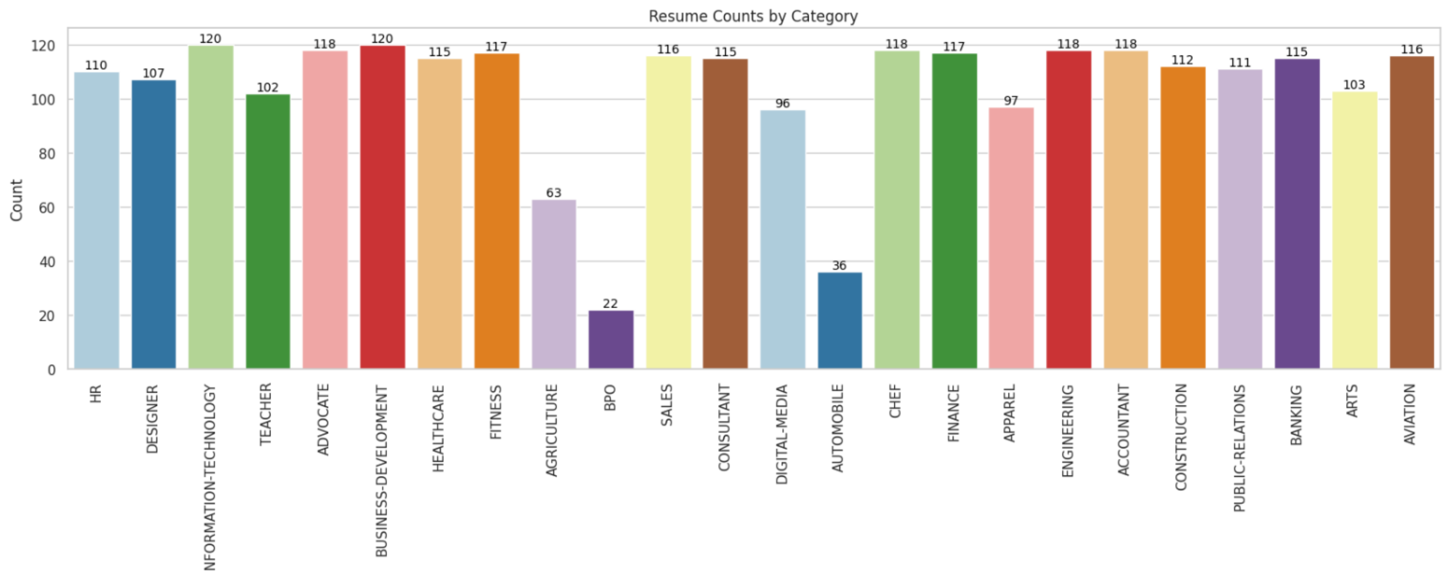


Figure 3.12: Resume Counts by Category

3.5.3 Bivariate Analysis (Job Postings)

This section explores the relationships between pairs of variables in the job postings data. The Salary Distribution by Experience Level box plot (Figure 3.13) revealed a clear positive correlation: median salaries consistently increased with experience level. 'Executive' and 'director' positions commanded the highest median salaries and the widest ranges, while 'entry level' and 'internship' roles had the lowest. A similar trend was observed in the Salary Distribution by Top 10 Industries box plot (Figure 3.14), with industries like 'software development' and 'IT services' having higher median salaries than sectors like 'retail' and 'hospitality'.

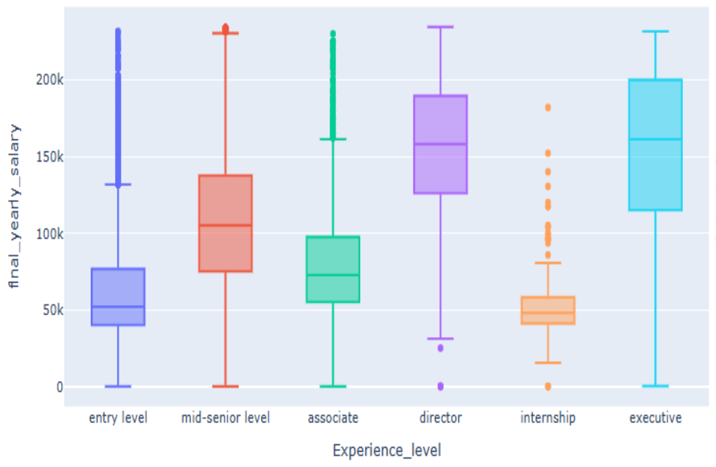


Figure 3.13: Annual Salary Distribution by Experience Level

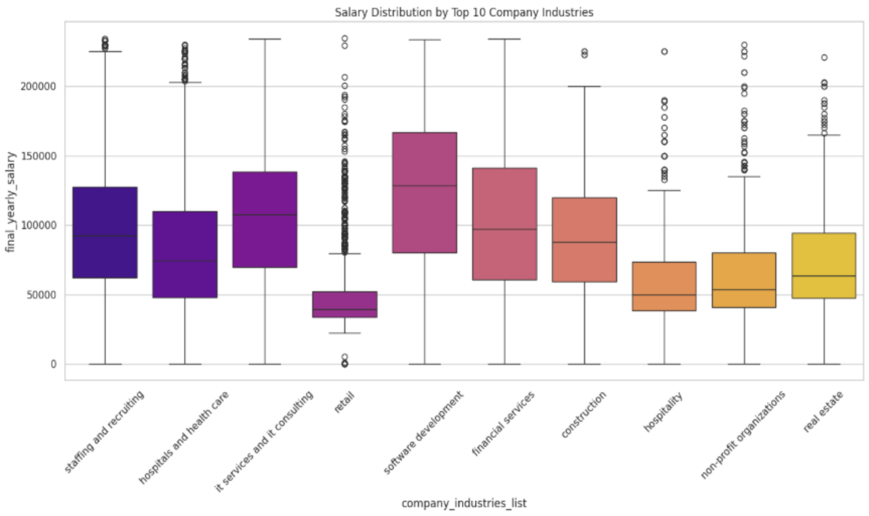


Figure 3.14: Salary Distribution by Top 10 Company Industries

An analysis of remote work revealed a distribution across different attributes. The Remote Allowed vs. Experience Level count plot (Figure 3.15) showed that while remote jobs existed across all levels, they were most prevalent in 'mid-senior' and 'entry level' roles. The Remote Allowed vs. Work Type plot (Figure 3.16) demonstrated that remote options were most common for 'full-time' positions, with significantly fewer remote opportunities for 'contract' and 'part-time' work.

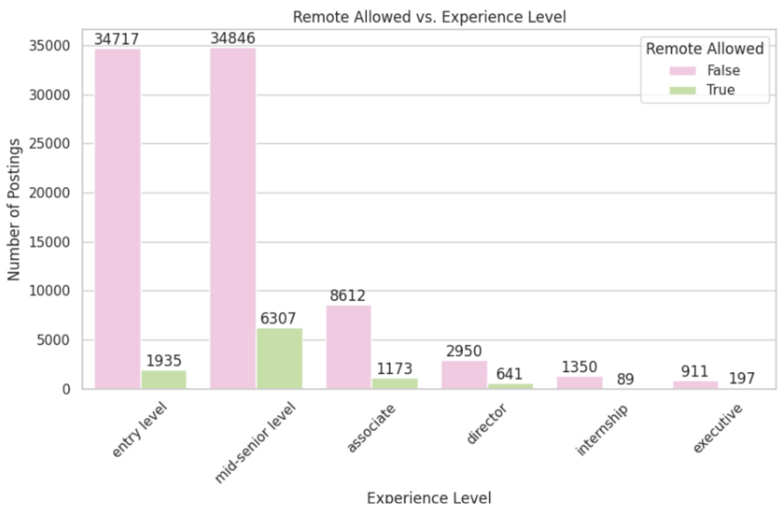


Figure 3.15: Remote Allowed vs. Experience Level

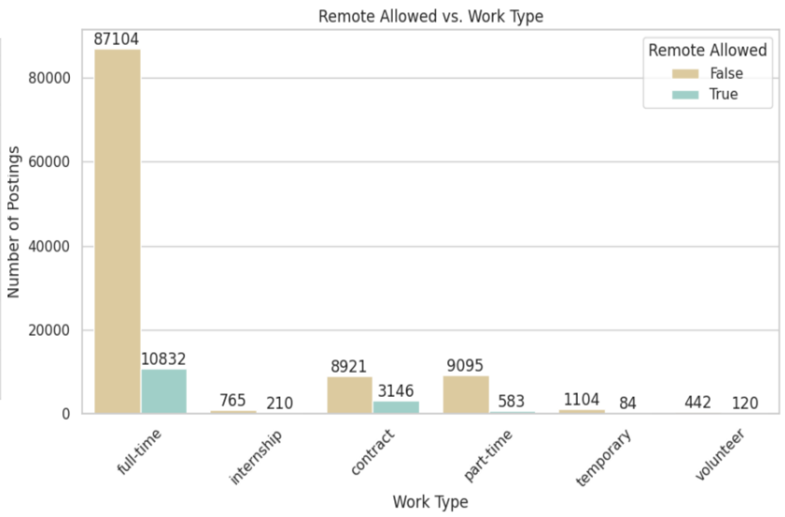


Figure 3.16: Remote Allowed vs. Work Type

A detailed look at specific roles showed that Median Salaries for the Top 10 Most Common Job Titles (Figure 3.17) varied significantly, with 'Project Manager' and 'Sales Manager' roles earning higher median salaries than 'Administrative Assistant' and 'Receptionist' positions.

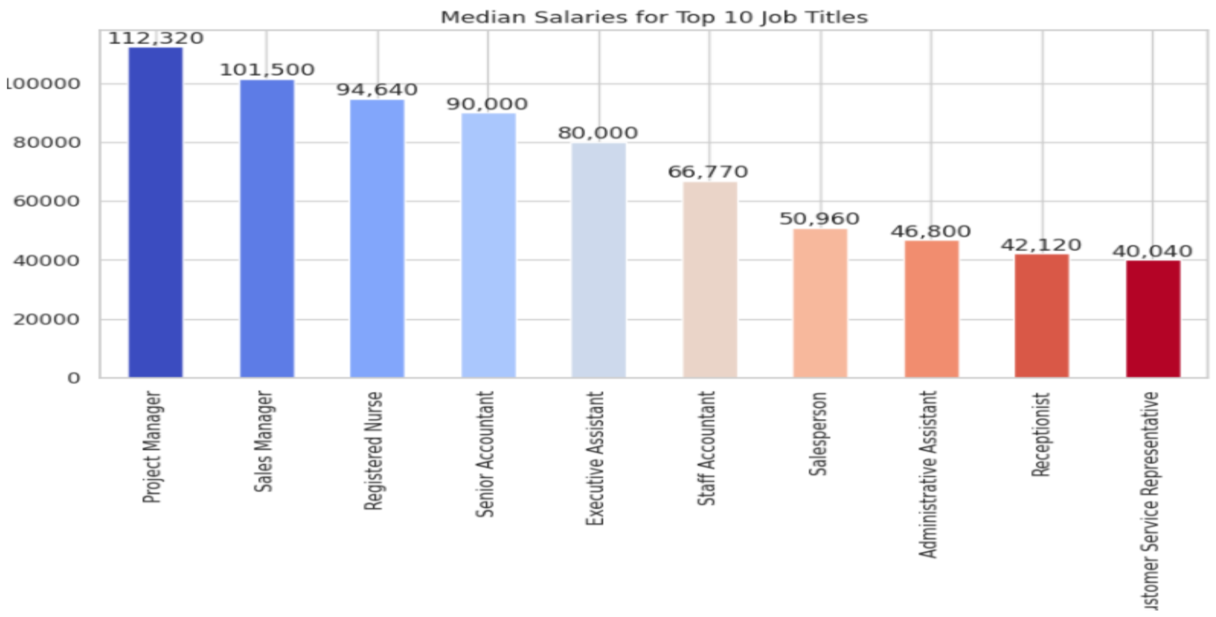


Figure 3.17: Median Salaries for Top 10 Most Common Job Titles

Finally, the heatmap of median salary by top 10 industries and experience level (Figure 3.18) provided a granular view of these relationships. It confirmed that median salaries generally increased with experience, with specific high-earning combinations found in the 'software development' and 'IT services' industries at the 'mid-senior', 'director', and 'executive' levels.

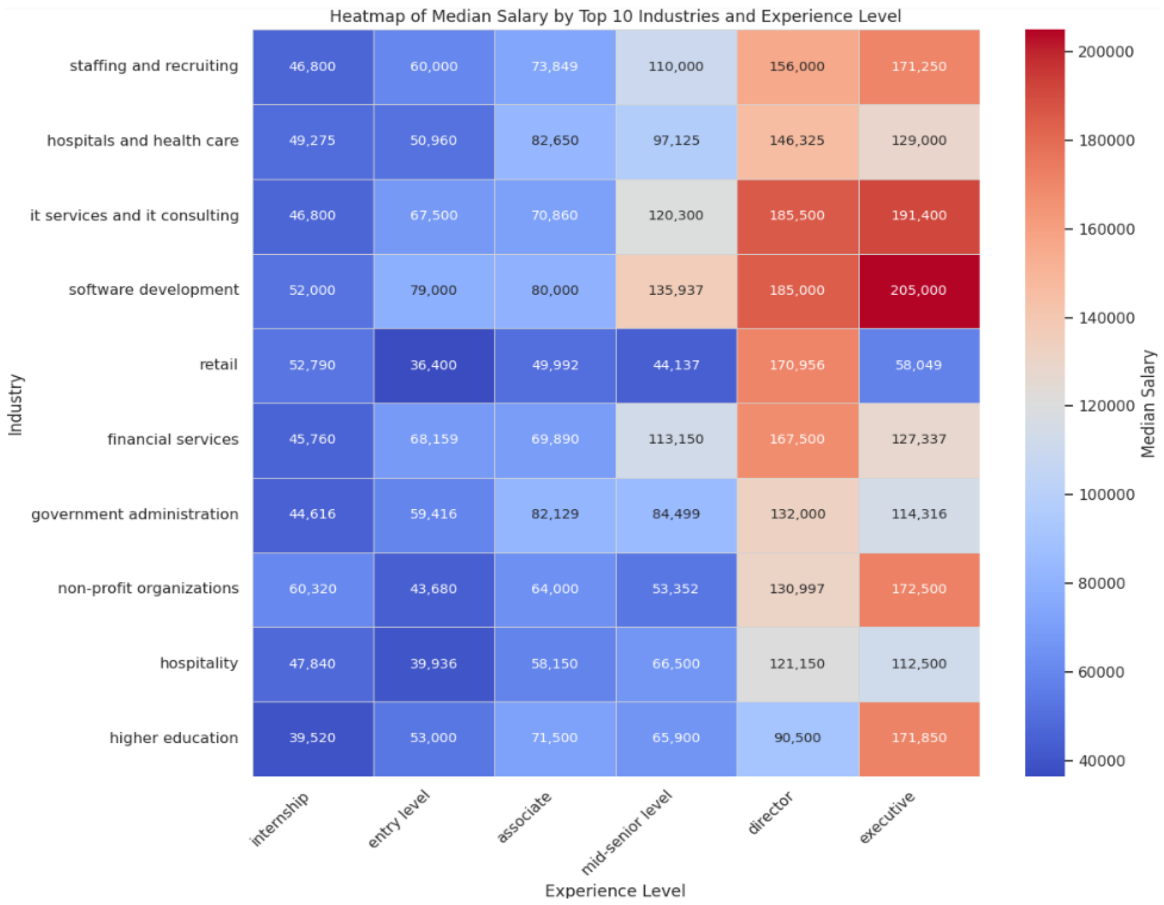


Figure 3.18: Heatmap of Median Salary by Top 10 Industries and Experience Level

3.6 Summary of Key Insights

The exploratory data analysis yielded several key insights critical for the subsequent development of the ResumeMatch platform. The datasets are large, diverse, and richly annotated with both textual and structured attributes, providing a robust foundation for analysis. There is ample information in both resumes and job descriptions to extract nuanced signals such as skills, seniority, and industry sector. The yearly salary feature, despite requiring significant conversion and cleaning, provided a consistent quantitative metric for comparisons and analysis of compensation trends. Furthermore, textual fields were effectively normalized to support future NLP operations, including Named Entity Recognition (NER) and embedding generation. Finally, the analysis confirmed that attributes such as remote work availability, industry sector, and job experience level are strong determinants of salary and overall job structure, underscoring their importance as features.

4. METHODOLOGY

This chapter presents a comprehensive overview of the technical architecture, experimental design, and toolchain employed in the development of the ResumeMatch system. The methodology is systematically organized into three distinct phases: Job Description Parsing, Resume Parsing, and Resume–Job Matching. This sequential approach ensured that each component could be independently developed, rigorously tested, and validated before its integration into the complete, end-to-end pipeline. The entire system was implemented in Python, utilizing a diverse set of libraries that collectively support natural language processing (NLP), machine learning, data manipulation, and visualization.

The primary technological stack includes pandas for data handling, NumPy for numerical operations, and scikit-learn for machine learning utilities. Core NLP tasks were managed by spaCy, NLTK, Gensim, and specialized libraries from Hugging Face Transformers. The neural network components were constructed using TensorFlow/Keras. Development and training were performed within a Google Colab environment, leveraging its integrated A100 GPU acceleration to expedite computationally intensive tasks. For post-processing, debugging, and data visualization, libraries such as Matplotlib, Plotly, and Streamlit were utilized.

4.1 Job Description Parsing

The initial and most critical phase of the project focused on the automated extraction of skills and other essential entities from job descriptions. The objective was to not only extract this information but also to perform a robust comparative analysis of multiple extraction techniques to identify the most effective method for the subsequent matching phases.

4.1.1 Evaluation Dataset Preparation

A meticulously prepared, annotated dataset was essential for training and benchmarking the various skill extraction models. A random subset of 250 job descriptions was selected from a pre-processed and cleaned corpus. From this subset, 150 samples were converted to a plain text format and meticulously annotated using Doccano for training a custom Named Entity Recognition (NER) model.

The remaining 100 samples from the 250-sample subset were reserved as a hold-out test set. To ensure an unbiased and reliable benchmark, the ground truth annotations for this test set, referred to as True_skills, were independently labeled using ChatGPT to assist in the model evaluation process. This approach provided a standardized and consistent basis for evaluating the performance of all extraction models, comparing their predictions against a single, reliable truth source.

4.1.2 Skill Database Construction

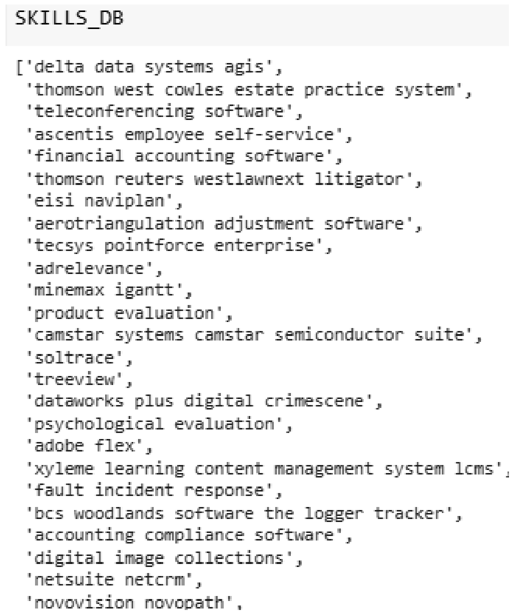
To underpin the performance of rule-based and embedding-based extraction methods, a comprehensive and robust skill database, termed SKILLS_DB, was constructed. This database was curated from three primary sources to ensure breadth and depth:

- **ONET Taxonomies:** The database incorporated structured skill taxonomies from the Occupational Information Network (ONET), specifically drawing from its Work Activities, Work

Styles, and Technology Skills categories. This provided a rich, standardized, and professionally recognized vocabulary of skills across a wide range of occupations.

- **Domain-Specific Mining:** We extracted 35 skills from our datasets, which are listed in skills.csv.
- **External Curation:** Supplementary lists were integrated from external, curated sources such as skillslabel.com. This step addressed potential gaps in the other two sources, particularly for underrepresented or highly specialized skills.

The final SKILLS_DB served as a foundational resource for matching-based extraction approaches, providing a rich vocabulary for identifying and categorizing skills and ensuring consistency across different methods while allowing for contextual flexibility.



```
SKILLS_DB

['delta data systems agis',
 'thomson west cowles estate practice system',
 'teleconferencing software',
 'ascentis employee self-service',
 'financial accounting software',
 'thomson reuters westlawnext litigator',
 'eisi naviplan',
 'aerotriangulation adjustment software',
 'tecsys pointforce enterprise',
 'adrelevance',
 'minemax igantt',
 'product evaluation',
 'camstar systems camstar semiconductor suite',
 'soltrace',
 'treeview',
 'dataworks plus digital crimescene',
 'psychological evaluation',
 'adobe flex',
 'xyleme learning content management system lcms',
 'fault incident response',
 'bcs woodlands software the logger tracker',
 'accounting compliance software',
 'digital image collections',
 'netsuite netcrm',
 'novovision novopath',
```

Figure 4.1: Skill Database Construction Overview

4.1.3 Implemented Skill Extraction Models

A diverse set of ten models was implemented and evaluated to determine the most effective strategy for skill and entity extraction. The performance of each model was measured against the True_skills test set using metrics such as Precision, Recall, F1-score, and Jaccard similarity. The models were grouped into two broad categories: Rule-Based & Embedding Methods and Model-Based Methods.

4.1.3.1 Rule-Based and Embedding-Based Methods

These methods are characterized by their reliance on predefined rules, patterns, or vector-based representations of text. They are generally computationally less expensive than model-based approaches.

- **Token Matching:** This method involved a straightforward linguistic approach. After removing common English stop words and punctuation, n-grams (unigrams, bigrams, and trigrams) were extracted from the job description text. These n-grams were then matched against the curated

SKILLS_DB. This approach is fast and effective for exact keyword matches but fails to capture skills that are semantically similar but phrased differently.

- **BERT Semantic Similarity:** To overcome the limitations of exact matching, this method leveraged a pre-trained Sentence-BERT model (all-MiniLM-L6-v2), which is highly optimized for generating dense vector embeddings for sentences and phrases. Embeddings were generated for all phrases within a job description and for all skills in the SKILLS_DB. Skill identification was based on a high cosine similarity score between a given phrase embedding and a skill embedding, indicating a close semantic relationship.

- **Combined Token Matching and Semantic Similarity:** This hybrid method was designed to capitalize on the strengths of both the token matching and semantic similarity approaches. The final set of extracted skills was the union of the results from both individual methods, providing a more comprehensive output that captured both exact keyword matches and semantically related skills.

- **PhraseMatcher (spaCy):** This approach utilized spaCy's highly optimized PhraseMatcher component. The SKILLS_DB was pre-processed into a list of matcher patterns, enabling extremely fast and efficient exact-phrase matching within the job description text.

- **YAKE (Yet Another Keyword Extractor):** As an unsupervised, statistical keyword extraction algorithm, YAKE was applied to the job descriptions. It identifies the most relevant keywords by analyzing various statistical features of the text, such as word position, frequency, and co-occurrence, without requiring any prior training data.

4.1.3.2 Model-Based Methods

These methods involve training or fine-tuning machine learning models to perform entity extraction, often resulting in higher accuracy and contextual awareness.

- **Transformer-Based Named Entity Recognition (NER) Models:** The study evaluated several pre-trained NER models from the Hugging Face repository, focusing on those specialized for skill recognition including algiraldohe/lm-ner-linkedin-skills-recognition, dslim/bert-base-NER, juro95/xlm-roberta-finetuned-ner-5-with-skills, Pot-l/bert-ner-skills, and jjzha/jobbert_skill_extraction. Ultimately, the analysis concentrated on two models for their distinct approaches:

    - algiraldohe/lm-ner-linkedin-skills-recognition: This model specializes in recognizing skills and business-related entities, classifying them into tags such as TECHNOLOGY, SOFT, and BUS. While effective for initial bootstrapping, it demonstrated limitations in generalizability across diverse domains.

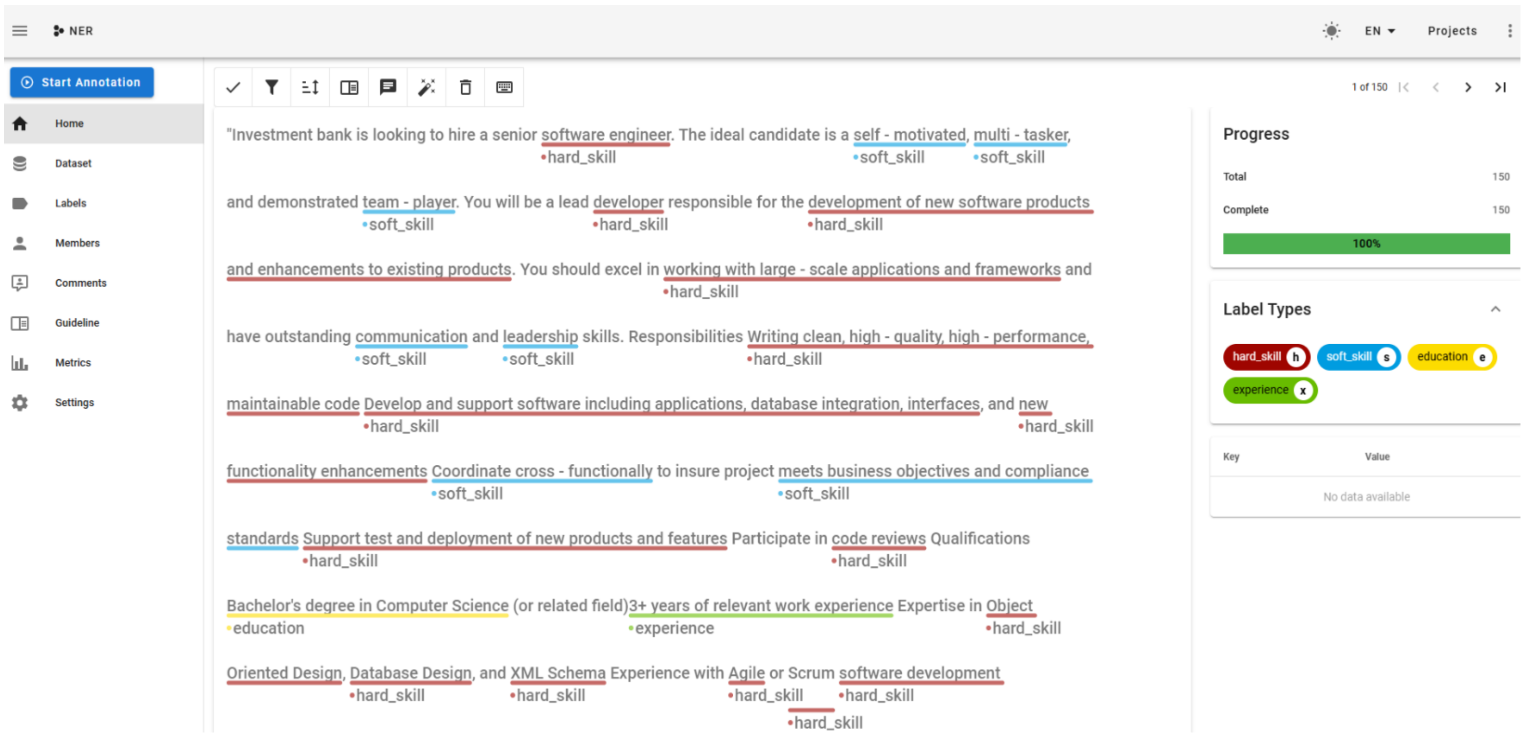
    - jjzha/jobbert_skill_extraction: This model provides highly granular, token-level extraction of soft skills and descriptive qualities. Its labels, however, did not always align with standard skill categories, complicating its use in a structured pipeline.

- **SkillNER:** This Python library leverages spaCy and a built-in skill taxonomy (based on Lightcast and ESCO) to extract and categorize skills. It is designed to capture nuanced skill mentions, providing both full matches and n-gram-scored candidates with relevance scores. The model successfully identified complex skills like "event planning" and "proactive," though this inclusivity could introduce noise requiring post-processing.
- **Nesta (ojd_daps_skills):** This is a dedicated skill extractor specifically designed for labor market data. Built on curated taxonomies, it uses a pre-trained NER model to map raw text onto a standardized skill list. This approach is highly scalable and demonstrated an excellent balance between structured output and contextual awareness, effectively capturing complex skill phrases and soft skills that are valuable for taxonomy-driven tasks. This library supports three different taxonomies for mapping: the European Commission's European Skills, Competences, and Occupations (ESCO), Lightcast's Open Skills, and a "toy" taxonomy for testing. For this research, the Lightcast taxonomy was specifically used.
- **LLM-Based Extraction:** This innovative method utilized a large language model (LLM), Google's Gemini 2.5 Flash Lite, accessed via the Gemini API. A carefully crafted structured prompt was used to instruct the LLM to extract and categorize hard skills, soft skills, education, and experience directly from the job description and return the results in a well-structured JSON format. This method consistently produced the most accurate and complete results by effectively handling nuanced phrasing and implicit skill mentions. However, its primary drawback was the token-based cost, which limits its scalability for a large-scale dataset.

4.1.3.3 Custom Named Entity Recognition (NER) Model Development

To extract structured information from unstructured job descriptions, a custom Named Entity Recognition (NER) model was developed. The model was trained to identify four key entity types- hard skills, soft skills, education, and experience which are critical for downstream tasks such as skill matching and resume-job alignment.

- **Dataset**
  A total of 150 job descriptions were manually annotated using Doccano, a web-based collaborative text annotation tool shown in figure 4.2. Each sample was tagged with entity spans corresponding to the four target categories. The annotated data was exported in .jsonl format, with each entry containing a text sample and its associated entity spans shown in figure 4.3. These spans were then validated using the BILUO tagging scheme to ensure proper alignment with token boundaries.



"Investment bank is looking to hire a senior software engineer (hard_skill). The ideal candidate is a self - motivated (soft_skill), multi - tasker (soft_skill),
and demonstrated team - player (soft_skill). You will be a lead developer (hard_skill) responsible for the development of new software products
and enhancements to existing products (hard_skill). You should excel in working with large - scale applications and frameworks (hard_skill) and
have outstanding communication (soft_skill) and leadership (soft_skill) skills. Responsibilities Writing clean, high - quality, high - performance,
maintainable code (hard_skill) Develop and support software including applications, database integration, interfaces (hard_skill), and new
functionality enhancements (hard_skill) Coordinate cross - functionally (soft_skill) to insure project meets business objectives and compliance
standards (soft_skill) Support test and deployment of new products and features (hard_skill) Participate in code reviews (hard_skill) Qualifications
Bachelor's degree in Computer Science (education) (or related field)3+ years of relevant work experience (experience) Expertise in Object
Oriented Design (hard_skill), Database Design (hard_skill), and XML Schema (hard_skill) Experience with Agile (hard_skill) or Scrum software development (hard_skill, hard_skill)

Figure 4.2: Doccano Interface

```
{"id":759,"text":"\"Investment bank is looking to hire a senior software engineer. The ideal candidate is a self - motivated, multi - tasker, and demonstrated team - player. You will be a
lead developer responsible for the development of new software products and enhancements to existing products. You should excel in working with large - scale applications and frameworks
and have outstanding communication and leadership skills. Responsibilities Writing clean, high - quality, high - performance, maintainable code Develop and support software including
applications, database integration, interfaces, and new functionality enhancements Coordinate cross - functionally to insure project meets business objectives and compliance standards
Support test and deployment of new products and features Participate in code reviews Qualifications Bachelor's degree in Computer Science (or related field)3+ years of relevant work
experience Expertise in Object Oriented Design, Database Design, and XML Schema Experience with Agile or Scrum software development methodologies Ability to multi - task, organize, and
prioritize work
\"","label":[[45,62,"hard_skill"],[89,105,"soft_skill"],[107,121,"soft_skill"],[140,153,"soft_skill"],[174,183,"hard_skill"],[204,278,"hard_skill"],[300,354,"hard_skill"],[376,389,"soft_sk
ill"],[394,404,"soft_skill"],[430,498,"hard_skill"],[499,584,"hard_skill"],[590,620,"hard_skill"],[621,652,"soft_skill"],[671,721,"soft_skill"],[722,778,"hard_skill"],[794,806,"hard_skill"
],[822,878,"education"],[878,914,"experience"],[928,950,"hard_skill"],[952,967,"hard_skill"],[973,983,"hard_skill"],[1000,1005,"hard_skill"],[1009,1014,"hard_skill"],[1015,1049,"hard_skill
"],[1061,1073,"soft_skill"],[1074,1083,"soft_skill"],[1089,1104,"soft_skill"]],"Comments":[]}
```

Figure 4.3: Annotated data (.jsonl) file format

- **Data Conversion and Splitting**

The annotated data was converted into spaCy's native binary format (DocBin) for efficient processing. The binary data was then randomly shuffled and partitioned into a training set and a development set using an 85/15 split ratio. This resulted in 127 documents for training and 23 for development.

- **Model Architecture and Configuration**

The custom NER model was developed using spaCy's transformer-based pipeline, specifically leveraging the en_core_web_trf model which is built on a RoBERTa-base transformer. The pipeline included a transformer component to generate contextualized token representations and an NER module for entity span classification. A configuration file (config.cfg) was generated using spaCy's init_config() utility to define the training parameters. The configuration specified English as the language, enabled GPU acceleration, and optimized the pipeline for accuracy, while disabling non-essential components to focus exclusively on the NER task.

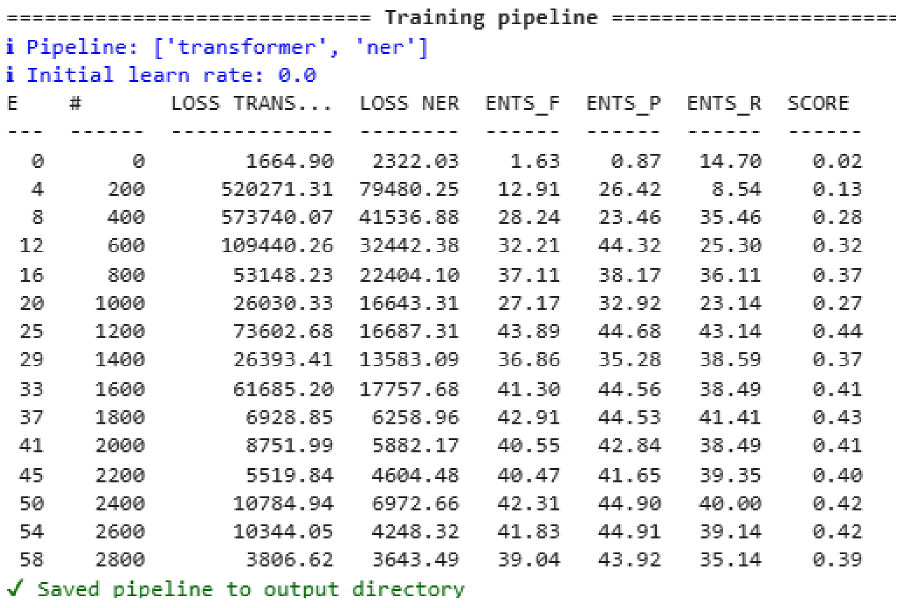
- **Training Procedure**

Training was conducted using the following command:

python -m spacy train config.cfg --output ./output --paths.train train.spacy --paths.dev dev.spacy --gpu-id 0

The model was fine-tuned over 60 epochs using a A100 GPU with a batch size of 8. The training command directed the model to use the preprocessed training and development datasets, logging performance metrics at every 200 steps. Key metrics tracked during training included the Entity-level F1 Score (ENTS_F), Precision (ENTS_P), and Recall (ENTS_R).

Upon completion, the final model achieved an F1 Score of 42.31, a Precision of 44.90, and a Recall of 40.00 on the development set. The trained pipeline was saved for downstream inference. This transformer-based architecture allows for a nuanced understanding of domain-specific language, and the annotated dataset ensures reliable supervision for accurate entity recognition.



```
============================ Training pipeline ======================
ℹ Pipeline: ['transformer', 'ner']
ℹ Initial learn rate: 0.0
E    #       LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  -------------  --------  ------  ------  ------  ------
  0       0        1664.90   2322.03    1.63    0.87   14.70    0.02
  4     200      520271.31  79480.25   12.91   26.42    8.54    0.13
  8     400      573740.07  41536.88   28.24   23.46   35.46    0.28
 12     600      109440.26  32442.38   32.21   44.32   25.30    0.32
 16     800       53148.23  22404.10   37.11   38.17   36.11    0.37
 20    1000       26030.33  16643.31   27.17   32.92   23.14    0.27
 25    1200       73602.68  16687.31   43.89   44.68   43.14    0.44
 29    1400       26393.41  13583.09   36.86   35.28   38.59    0.37
 33    1600       61685.20  17757.68   41.30   44.56   38.49    0.41
 37    1800        6928.85   6258.96   42.91   44.53   41.41    0.43
 41    2000        8751.99   5882.17   40.55   42.84   38.49    0.41
 45    2200        5519.84   4604.48   40.47   41.65   39.35    0.40
 50    2400       10784.94   6972.66   42.31   44.90   40.00    0.42
 54    2600       10344.05   4248.32   41.83   44.91   39.14    0.42
 58    2800        3806.62   3643.49   39.04   43.92   35.14    0.39
✓ Saved pipeline to output directory
```

Figure 4.4: Custom NER Training pipeline

4.2 Resume Parsing

Parsing resumes presents a significantly greater challenge than parsing job descriptions due to the high degree of variability in document structure and formatting. Resumes often contain non-standard headings, multi-column layouts, embedded tables, and a wide array of stylistic designs, which can make traditional rule-based methods ineffective. Three distinct methods were evaluated for extracting structured information from a dataset of 2,482 resumes.

4.2.1 PyResparser

PyResparser is a simple, rule-based tool designed for extracting common information from resumes in PDF and DOCx formats. Its extraction capabilities include name, email, mobile numbers, skills, total experience, college name, degree, designation, and company names. The implementation involves initializing the ResumeParser object with the path to the resume file and calling the get_extracted_data() method. The output is a dictionary containing the extracted fields.

4.2.2 Custom NER Model

The custom NER model which we trained specifically trained on job descriptions, was applied to the resume dataset to assess its generalizability. Despite not being trained on resume-specific data, the model demonstrated a surprising ability to capture contextually appropriate hard and soft skills, as well as educational credentials. While the model sometimes extracted fragmented or overly long phrases, the overall quality of its structured output was reliable, validating its potential for cross-document-type application.

4.2.3 LLM-Based Resume Parsing

This method, mirroring the approach used for job descriptions, leveraged Google's Gemini 2.5 Flash Lite API with specially designed prompts. The prompts were tailored to instruct the LLM to extract structured resume data, including hard skills, soft skills, education, and experience, into a JSON format. This approach proved to be the most accurate and context-aware, consistently producing well-organized and correctly categorized outputs, which are indispensable for effective resume-job matching.

4.3 Resume-Job Matching and Ranking

The final phase of the methodology involved developing and evaluating multiple semantic similarity approaches to compute a matching score between job descriptions and resumes. The core objective was to identify and rank the most relevant resumes for a given job and vice versa, providing a bidirectional ranking system.

The system was designed to perform a bidirectional match:

- **Job-to-Resume:** Identify the top 5 most suitable candidates for each job description.
- **Resume-to-Job:** Identify the top 5 most suitable jobs for each resume.

The similarity scores were aggregated at a section level using a weighted average. Each resume and job description were represented by four key sections, each with a predetermined weight to reflect its relative importance in the matching process:

- **Hard Skills:** Weight = 0.4
- **Soft Skills:** Weight = 0.2
- **Education:** Weight = 0.2
- **Experience:** Weight = 0.2

For each section, cosine similarity was computed between the job and resume vectors. These section-specific scores were then aggregated using the defined weights to produce a final, comprehensive matching score.

Four distinct embedding approaches were implemented and evaluated to determine the most effective representation for calculating semantic similarity.

4.3.1 TF-IDF

This method applied TF-IDF (Term Frequency-Inverse Document Frequency) vectorization using unigrams and bigrams, with a maximum feature limit of 5,000. Cosine similarity was then used to measure the textual overlap between documents. While this approach is computationally fast and the results are easily interpretable, it is limited to surface-level lexical matching and lacks the ability to capture deeper semantic relationships.

4.3.2 Latent Dirichlet Allocation (LDA)

This topic modeling approach represented each document (resume or job description) as a probability distribution over 10 latent topics. Thematic similarity was then measured by computing cosine similarity in this topic space.

4.3.3 Word2Vec

A pretrained Word2Vec model (word2vec-google-news-300) was used on the combined corpus of job descriptions and resumes. Each document was then represented by the average of the word embeddings within it. Cosine similarity on these average embeddings provided a more nuanced measure of thematic alignment than TF-IDF.

4.3.4 BERT Embeddings

This approach used BERT-based embeddings, specifically from the all-MiniLM-L6-v2 Sentence Transformer. This model generated dense, contextualized vectors for each document, which captured complex semantic and syntactic nuances.

4.4 Prototype UI (Streamlit)

The two user flows were designed to cater to both job seekers and recruiters. The Job Seeker flow allows users to upload their resume and a job description. The system then displays a bar plot showing semantic, exact, and partial match scores, a weighted final score, and a list of five suggested interview questions based on the job description. The Recruiter flow allows a user to upload a single job description and multiple resumes. The system then clusters the resumes, calculates the job description-to-cluster similarity, and presents a ranking table of candidates with expandable skill gap analysis cards.

Semantic Match: Compares the conceptual meaning of two texts using embedding-based similarity rather than exact string equality. Exact Match: Checks for precise character-for-character equivalence between two text strings. Partial (Fuzzy) Match: Computes an approximate similarity score by allowing minor variations like typos or word reordering.

Key interactive controls for the UI include a section weights slider, a threshold slider for fuzzy matching, a matching mode toggle (Semantic / Exact / Partial), and a cluster K slider.

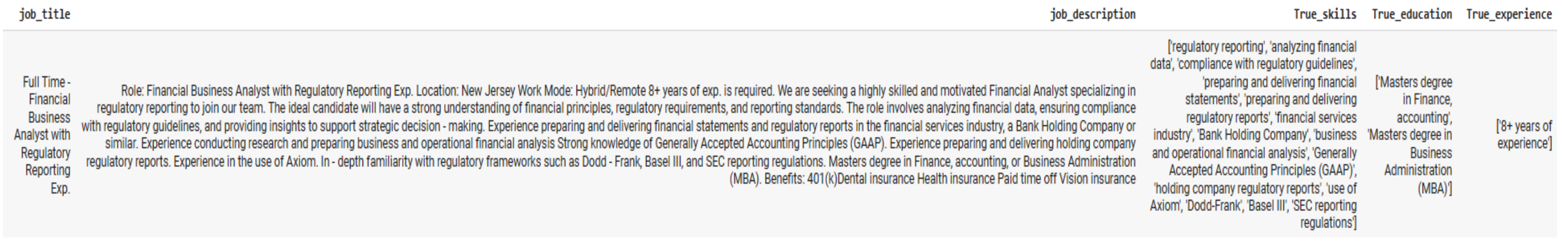
5. RESULTS

This chapter presents a comprehensive overview of the experimental results obtained from each skill extraction approach implemented in our research. It is divided into three experimental stages: job description parsing (skill extraction), resume parsing, and resume-job matching.

5.1 Job Description Parsing: Skill Extraction Approaches

To provide a comprehensive assessment of the skill extraction pipeline, rigorous performance evaluation was conducted using standard metrics Precision, Recall, F1-score, and Jaccard Accuracy.

1. Ground Truth Labels: A hold-out set of 100 job descriptions was annotated with ground truth labels for True skills, True education, and True experience. These annotations served as the benchmark for evaluating the performance of each model. While all models were primarily evaluated on True skills, a more granular comparison of True education and True experience was conducted for the top-performing models in a subsequent analysis.



job_title	job_description	True_skills	True_education	True_experience
Full Time - Financial Business Analyst with Regulatory Reporting Exp.	Role: Financial Business Analyst with Regulatory Reporting Exp. Location: New Jersey Work Mode: Hybrid/Remote 8+ years of exp. is required. We are seeking a highly skilled and motivated Financial Analyst specializing in regulatory reporting to join our team. The ideal candidate will have a strong understanding of financial principles, regulatory requirements, and reporting standards. The role involves analyzing financial data, ensuring compliance with regulatory guidelines, and providing insights to support strategic decision - making. Experience preparing and delivering financial statements and regulatory reports in the financial services industry, a Bank Holding Company or similar. Experience conducting research and preparing business and operational financial analysis Strong knowledge of Generally Accepted Accounting Principles (GAAP). Experience preparing and delivering holding company regulatory reports. Experience in the use of Axiom. In - depth familiarity with regulatory frameworks such as Dodd - Frank, Basel III, and SEC reporting regulations. Masters degree in Finance, accounting, or Business Administration (MBA). Benefits: 401(k)Dental insurance Health insurance Paid time off Vision insurance	['regulatory reporting', 'analyzing financial data', 'compliance with regulatory guidelines', 'preparing and delivering financial statements', 'preparing and delivering regulatory reports', 'financial services industry', 'Bank Holding Company', 'business and operational financial analysis', 'Generally Accepted Accounting Principles (GAAP)', 'holding company regulatory reports', 'use of Axiom', 'Dodd-Frank', 'Basel III', 'SEC reporting regulations']	['Masters degree in Finance, accounting', 'Masters degree in Business Administration (MBA)']	['8+ years of experience']

Figure 5.1: Example of Ground Truth Labels for an Annotated Job Description

2. Evaluation Metrics: The following metrics were used to quantify the accuracy of the extracted entities:

   - Precision: The proportion of correctly identified skills among all extracted skills.
   - Recall: The proportion of truly relevant skills that were successfully identified by the method.
   - F1-score: The harmonic mean of precision and recall, offering a balanced measure of overall accuracy.
   - Jaccard Accuracy: A measure of the similarity between the set of extracted skills and the set of ground truth skills, calculated as the intersection over the union of the predicted and true sets.

To accommodate minor lexical variations between extracted entities and ground-truth labels, fuzzy string matching (RapidFuzz) was used with a similarity threshold of 80%; this adjustment was applied consistently across all models to produce robust, realistic performance estimates. The approaches are grouped into two categories: (a) Rule-based & embedding methods and (b) Model-based methods. The results are shown below.

5.1.1 Rule-Based & Embedding Methods

These approaches rely on exact or statistical matching techniques, ranging from n-gram lookups to vector-based similarity, to extract skills without supervised learning.

- **Token Matching (Unigrams, Bigrams)**
  This method achieved a Precision of 0.450, Recall of 0.154, F1-score of 0.214, and Jaccard Accuracy of 0.128. Its strength lies in being extremely fast and easy to implement, but it fails to recognize synonyms or varied phrasing and depends entirely on how comprehensive the SKILLS_DB is, resulting in many missed skills.
- **BERT Embeddings (Semantic Matching)**
  By using a pretrained BERT-based Sentence Transformer for semantic similarity, this method achieved a Precision of 0.390, Recall of 0.217, F1-score of 0.260, and Jaccard Accuracy of 0.157. While it can catch semantically related terms, its performance is sensitive to the similarity threshold and embedding quality, which can produce false positives.
- **Combined Token + Embedding Matching**
  This hybrid method, the union of results from both methods, yielded a Precision of 0.441, Recall of 0.168, F1-score of 0.229, and Jaccard Accuracy of 0.137. It captures more skills than token matching alone, but combining the methods only partially offsets their individual weaknesses.
- **SpaCy PhraseMatcher**
  This approach achieved a Precision of 0.419, Recall of 0.160, F1-score of 0.218, and Jaccard Accuracy of 0.130. Like other exact-match methods, it cannot detect synonyms or skills phrased differently, leading to low recall.
- **YAKE (Yet Another Keyword Extractor)**
  The unsupervised keyword extraction algorithm, YAKE, exhibited low performance for this task, with a Precision of 0.218, Recall of 0.211, F1-score of 0.184, and Jaccard Accuracy of 0.106. YAKE's ability to identify important terms that might represent skills without relying on a predefined skill database is a strength. However, it is not well-suited for precise skill extraction without substantial post-processing in our experimental setup, as it often extracts non-skill terms.

5.1.2 Model-Based Methods

The model-based category comprises pretrained transformer NERs, domain-specific toolkits, LLM-based extractors, and a custom NER model fine-tuned on annotated job descriptions. These approaches leverage contextual information and learned patterns to identify and categorise skill-like entities.

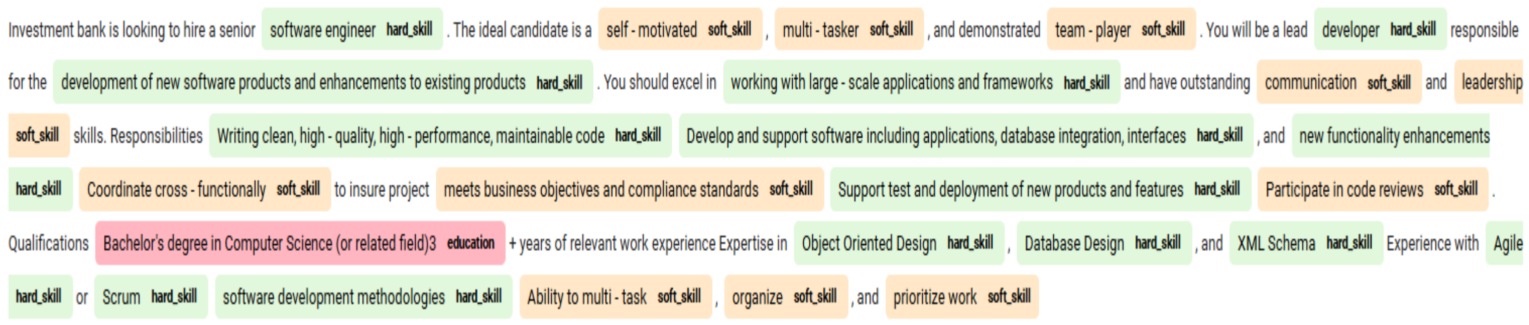
- **Transformer-Based Named Entity Recognition (NER) Models**
  The final evaluation emphasizes two models: algiraldohe/lm-ner-linkedin-skills-recognition and jjzha/jobbert_skill_extraction due to their domain-specific orientations. The algiraldohe model, fine-tuned on LinkedIn data, achieved Precision 0.458, Recall 0.269, F1-score 0.308, and Jaccard Accuracy 0.196. Its domain training aids in categorizing technology and soft skills, but its results may not generalize perfectly to all job descriptions. The jjzha/jobbert_skill_extraction model, which specifically targeted soft skills and descriptive qualities, achieved competitive metrics among the initial transformer models, with Precision 0.467, Recall 0.299, F1-score 0.328, and Jaccard Accuracy of 0.237. This model's focus on a broader definition of 'skills' is a notable advantage. However, the merging process and specific focus sometimes led to the extraction of phrases not typically

considered discrete "skills" within a standard taxonomy. The lower precision 0.467 observed suggests a potential for false positives.

- **SkillNER**
  This Python library demonstrated a good balance between precision 0.417 and recall 0.471, resulting in an F1-score of 0.409 and Jaccard Accuracy of 0.269. A key strength of SkillNER is its specific design for skill extraction and its utilization of a pre-built taxonomy. It effectively combines different matching strategies, but its performance is inherently tied to the quality and coverage of its built-in skill taxonomy.
- **ojd_daps_skills (Nesta Skill Extraction)**
  Leveraging the ojd_daps_skills library with the Lightcast taxonomy, this method achieved a Precision of 0.530, Recall of 0.506, F1-score of 0.489, and Jaccard Accuracy of 0.345, demonstrating a strong balance. However, its accuracy depends on the underlying model and the specific taxonomy, which may not cover emerging skills.
- **API-Accessed LLM-Based Extraction (Google Gemini 2.5 Flash Lite)**
  This method demonstrated the highest Precision of 0.654, Recall of 0.502, F1-score of 0.528, and Jaccard Accuracy of 0.387. A key strength of LLMs is their ability to understand context and nuances in text, potentially leading to higher accuracy in extracting skills and other entities and handling linguistic variations, while providing structured output. However, performance can be sensitive to prompt design, and API usage may raise concerns regarding data privacy, cost, or rate limits.

5.1.3 Custom Named Entity Recognition (NER)

We developed a custom Named Entity Recognition model fine-tuned on 150 manually annotated job descriptions. This model achieved the highest Recall (0.725) and F1-score (0.621), a Precision of 0.613, and a Jaccard Accuracy of 0.476 for skill extraction among all methods evaluated in this study. A significant strength of a custom NER model is its high effectiveness in capturing relevant skills from job descriptions. Its performance is notably influenced by the size and quality of the training data (150 samples), which may underrepresent rare or newly emerging terms.



Investment bank is looking to hire a senior software engineer hard_skill . The ideal candidate is a self - motivated soft_skill , multi - tasker soft_skill , and demonstrated team - player soft_skill . You will be a lead developer hard_skill responsible for the development of new software products and enhancements to existing products hard_skill . You should excel in working with large - scale applications and frameworks hard_skill and have outstanding communication soft_skill and leadership soft_skill skills. Responsibilities Writing clean, high - quality, high - performance, maintainable code hard_skill Develop and support software including applications, database integration, interfaces hard_skill , and new functionality enhancements hard_skill Coordinate cross - functionally soft_skill to insure project meets business objectives and compliance standards soft_skill Support test and deployment of new products and features hard_skill Participate in code reviews soft_skill . Qualifications Bachelor's degree in Computer Science (or related field)3 education + years of relevant work experience Expertise in Object Oriented Design hard_skill , Database Design hard_skill , and XML Schema hard_skill Experience with Agile hard_skill or Scrum hard_skill software development methodologies hard_skill Ability to multi - task soft_skill , organize soft_skill , and prioritize work soft_skill

Figure 5.2: Example of Color-Mapped Custom NER output

5.1.4 Overall Evaluation Metrics & Comparison (Job Description Parsing)

Table 5.1 presents the aggregate performance metrics across all evaluated methods:

Model	Precision	Recall	F1 Score	Jaccard Accuracy
tokenized_skills	0.450	0.154	0.214	0.128
BERT_Semantic_skills	0.390	0.217	0.260	0.157
Token_BERT_combined_skills	0.441	0.168	0.229	0.137
algiraldohe_skills_lm_linkedin_ner	0.458	0.269	0.308	0.195
jobbert_skill_extraction	0.467	0.299	0.328	0.232
spaCy PhraseMatcher	0.419	0.160	0.218	0.130
SkillNER	0.417	0.471	0.409	0.269
YAKE	0.218	0.211	0.184	0.106
ojd_daps_skills (Nesta)	0.530	0.506	0.489	0.345
LLM (Google Gemini 2.5 Flash Lite)	0.654	0.502	0.528	0.387
custom_ner_skills	0.613	0.725	0.621	0.476

Table 5.1: Skill Extraction Model Performance Metrics

Analyzing the individual model metrics reveals a range of performance characteristics and trade-offs. The Custom NER model achieved the highest F1-score 0.621 and Jaccard Accuracy 0.476, indicating it effectively balances precision and recall. The LLM was the second-best performing method based on F1-score 0.528, then ojd_daps_skills F1-score 0.489 also showed respectable performance. The trade-offs between precision and recall were evident. Methods prioritizing exact matches or broad keyword identification tended to have skewed performance. More sophisticated models, particularly those with domain-specific training or powerful language understanding capabilities, achieved a better balance.

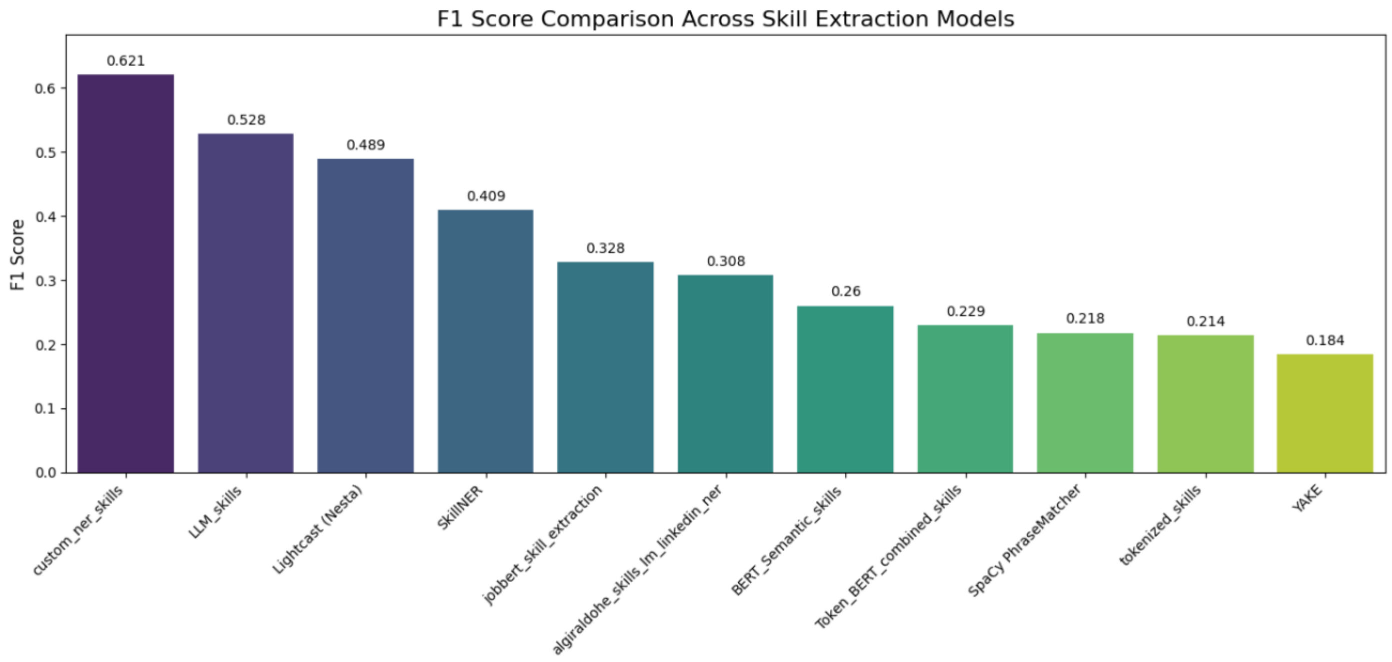


Figure 5.3: F1-Score Comparison Across Skill Extraction Models

5.1.5 Top Model Comparison: Custom NER vs. LLM-Based Extraction (Job Description Parsing)

In this research, the Custom NER model and the LLM-based Extraction method emerged as top performers for skill extraction. A side-by-side comparison of their performance metrics for each entity type (skills, education, and experience) is shown in Table 5.2

Entity	Model	Precision	Recall	F1-score	Jaccard Accuracy
Skills	Custom NER	0.615	0.727	0.623	0.478
	LLM	0.654	0.502	0.528	0.387
Education	Custom NER	0.461	0.430	0.429	0.387
	LLM	0.530	0.545	0.516	0.471
Experience	Custom NER	0.629	0.579	0.571	0.507
	LLM	0.538	0.642	0.551	0.477

Table 5.2: Entity-Wise Comparison (Custom NER vs. LLM)

Analyzing these results, we observe the Custom NER model achieved the strongest overall performance for skills extraction, outperforming other methods on both F1 and Jaccard similarity. However, the

LLM-based approach demonstrated superior accuracy for Education and achieved higher recall for Experience, while the Custom NER retained the advantage in F1 and Jaccard for Experience.

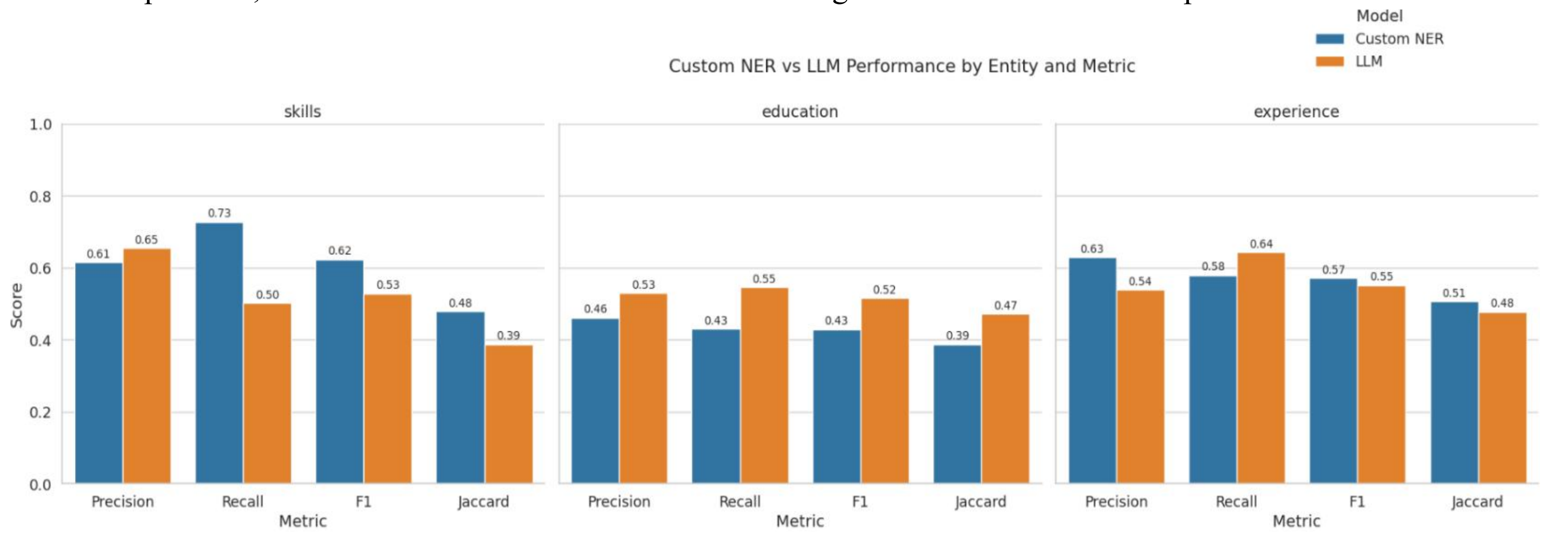


Figure 5.4: Custom NER vs LLM Performance by Entity Type

5.2 Resume Parsing Approaches

This section presents the results of three resume parsing approaches evaluated on the resume dataset (2,482 resumes) through a qualitative assessment of each method's ability to extract four key entity types: Education, Experience, Hard Skills, and Soft Skills.

5.2.1 PyResparser (Rule-Based Extraction)

Execution demonstrated PyResparser's ability to extract basic fields, but outputs contained inconsistencies and misclassifications. These results highlight the dependency of rule-based parsers on consistent formatting; when resume layouts deviate, extractions become noisy and incomplete (for example, company names and locations were sometimes included within the extracted experience text).

5.2.2 Custom NER Model

This approach utilizes our fine-tuned Custom Named Entity Recognition (NER) model. The process involves loading a pre-trained custom NER model from a zip file. Applying the custom NER model to a subset of the resume dataset successfully extracted entities categorized as hard skills, soft skills, education, and experience. This method provides a strong balance between precision and adaptability and is especially valuable due to its full retainability and local deployment capability without reliance on external APIs.

5.2.3 LLM-Based Resume Parsing with Gemini 2.5 Flash-lite API

This approach leveraged Google's Gemini 2.5 Flash-lite to parse resume text into a predefined JSON structure. It successfully identified educational qualifications, experiences, and categorized skills. The LLM demonstrated the ability to extract specific details and categorize skills based on the prompt's instructions. The output format is clean and structured, making it easy to process. The major limitations are operational: the API is subject to rate limits, usage costs, and a lack of transparency over internal logic, which hinder its scalability in production environments.

5.2.4 Overall Comparison of Resume Parsing Approaches

Table 5.3 provides a comparative analysis of the three resume parsing methods: PyResparser, spaCy Custom NER, and LLM-Based Resume Parsing. The evaluation considers entity coverage, contextual accuracy, domain adaptability, and practical deployment factors.

Criterion	PyResparser	SpaCy Custom NER	LLM-Based API (Gemini 2.5)
Entity Coverage	Moderate (skills, job titles)	High (skills, soft skills, education, experience)	Very High (comprehensive, structured fields)
Context Awareness	Low (rule-based surface patterns)	Moderate to High (context-aware, learned entity boundaries)	High (in-depth sentence-level semantic reasoning)
Extraction Quality	Inconsistent (duplicates, noise)	Accurate with some fragmented phrases	High (concise and coherent spans, minor redundancy)
Domain Adaptability	Low	High (trained on job descriptions, performs well on resumes)	Low (general-purpose, not domain-tuned)
Customization	None	High (retrainable on new data, label control)	None (black-box, fixed behavior)
Cost and Deployment	Free, fast, low resource	Free, runs locally after training	Costly, rate-limited, dependent on external API

Table 5.3: Overall Comparison of Resume Parsing Approaches

The LLM-based resume parsing achieves strong contextual understanding and high entity coverage, but it is limited by API cost and scalability challenges. In contrast, the Custom NER model, despite being trained on job descriptions, demonstrates remarkably strong generalization to resumes. It captures nuanced skills, experience summaries, and educational background with high semantic alignment. The PyResparser approach, while simple, lacks contextual sensitivity and often produces incomplete extractions.

5.3 Resume-Job Matching and Ranking

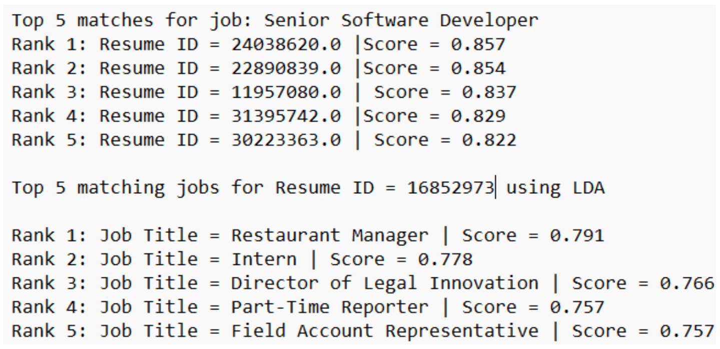
This section describes the design and evaluation of embedding-based approaches for computing semantic similarity between resumes and job descriptions. The matching process was based on structured content extracted in earlier stages hard skills, soft skills, education, and experience. Four embedding techniques were implemented and compared:

Method	Description
TF-IDF	Traditional sparse vector representation using term frequency-inverse document frequency
LDA	Latent Dirichlet Allocation-based topic distributions as document vectors
Word2Vec	Average of word embeddings using pre-trained Google News vectors

BERT	Contextualized sentence embeddings using all-MiniLM-L6-v2 via Sentence-BERT

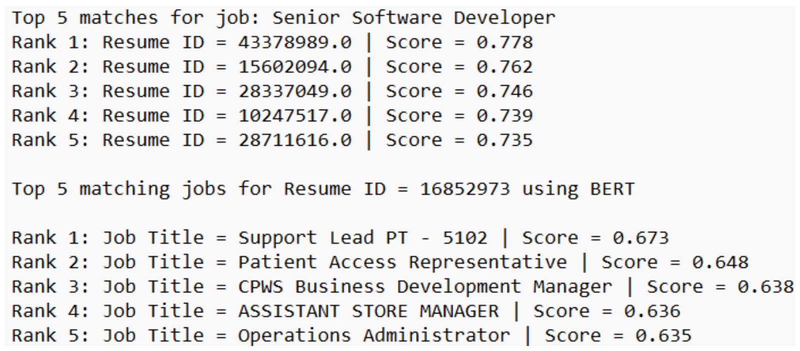
Table 5.4: Embedding Approaches for Resume-Job Matching

The similarity scores were used to generate ranked outputs: the top five resumes for a given job and the top five jobs for a given resume. The quality of these rankings was assessed using unsupervised evaluation metrics to compare the performance of the embedding methods. The objective was to determine which technique most effectively captured relevant relationships between resumes and job descriptions, thereby providing reliable and interpretable recommendations for recruitment use cases.



```
Top 5 matches for job: Senior Software Developer
Rank 1: Resume ID = 24038620.0 |Score = 0.857
Rank 2: Resume ID = 22890839.0 |Score = 0.854
Rank 3: Resume ID = 11957080.0 | Score = 0.837
Rank 4: Resume ID = 31395742.0 |Score = 0.829
Rank 5: Resume ID = 30223363.0 | Score = 0.822

Top 5 matching jobs for Resume ID = 16852973 using LDA

Rank 1: Job Title = Restaurant Manager | Score = 0.791
Rank 2: Job Title = Intern | Score = 0.778
Rank 3: Job Title = Director of Legal Innovation | Score = 0.766
Rank 4: Job Title = Part-Time Reporter | Score = 0.757
Rank 5: Job Title = Field Account Representative | Score = 0.757
```

Figure 5.5: LDA Top 5 Resume-Job Matching and Ranking



```
Top 5 matches for job: Senior Software Developer
Rank 1: Resume ID = 43378989.0 | Score = 0.778
Rank 2: Resume ID = 15602094.0 | Score = 0.762
Rank 3: Resume ID = 28337049.0 | Score = 0.746
Rank 4: Resume ID = 10247517.0 | Score = 0.739
Rank 5: Resume ID = 28711616.0 | Score = 0.735

Top 5 matching jobs for Resume ID = 16852973 using BERT

Rank 1: Job Title = Support Lead PT - 5102 | Score = 0.673
Rank 2: Job Title = Patient Access Representative | Score = 0.648
Rank 3: Job Title = CPWS Business Development Manager | Score = 0.638
Rank 4: Job Title = ASSISTANT STORE MANAGER | Score = 0.636
Rank 5: Job Title = Operations Administrator | Score = 0.635
```

Figure 5.6: BERT Top 5 Resume-Job Matching and Ranking

5.3.1 Separation@K Evaluation

The Separation@K metric was applied to evaluate each embedding method's ability to distinguish between highly relevant and less relevant resume matches for a given job posting. Larger Separation@K values indicate stronger discrimination between relevant and irrelevant matches. Separation@K was calculated for two different values of K: 5 and 10. The results, summarized in the table 5.5, demonstrate a clear hierarchy in performance among the tested methods:

Method	Sep@5	Sep@10
LDA	0.9586	0.9543
BERT	0.6062	0.5735
Word2Vec	0.5458	0.5322
TF-IDF	0.2297	0.2052

Table 5.5: Separation@K Evaluation Results

LDA achieved the highest Separation@K scores for both K=5 (0.9586) and K=10 (0.9543), indicating a significantly larger gap between the average similarity scores of the top-ranked and bottom-ranked resumes. The Separation@K curve, presented in Figure 5.7, reinforces these findings.

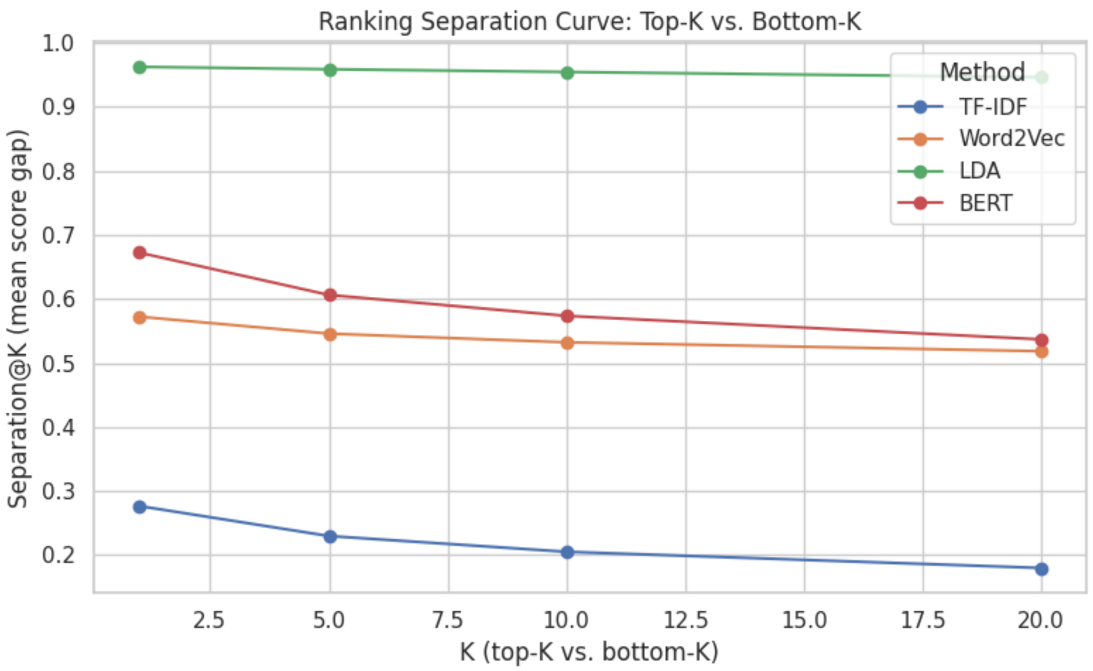


Figure 5.7: Ranking Separation Curve: Top-K vs. Bottom-K

5.3.2 Unsupervised Metrics (Cluster Agreement, Reciprocal Top-K, Mutual Rank)

In addition to the Separation@K analysis, a suite of unsupervised proxy metrics was employed to further evaluate the quality of the job-resume matchings generated by each embedding method. These metrics provide insights into different aspects of the matching performance without requiring explicit ground truth labels. The three metrics utilized were Cluster Agreement, Reciprocal Top-K Agreement, and Mutual Rank Score.

- Cluster Agreement: This metric assesses the degree to which the top-K matched resumes for a given job belong to the same cluster as the job itself, based on clustering performed on the combined embeddings. A higher score means better agreement between the embedding space clustering and the top-K job-resume matches.
- Reciprocal Top-K Agreement: This metric quantifies the extent of mutual overlap between the set of top-K resumes for a given job and the set of top-K jobs for a given resume. A higher Reciprocal Agreement score suggests a stronger reciprocal relationship in the top rankings, indicating that if a job is highly ranked for a resume, that resume is also likely to be highly ranked for that job.
- Mutual Rank Score: This metric calculates the average rank of mutually top-ranked job-resume pairs. A lower score indicates that mutually high-ranking pairs are assigned better (lower) ranks on average.

The results for these unsupervised proxy metrics are summarized in the table 5.6:

Model	Cluster Agree	Reciprocal	Mutual Rank
LDA	0.9368	0.7272	7.6976
BERT	0.9268	0.7716	5.9256
TF-IDF	0.4936	0.5644	29.3272
Word2Vec	0.5608	0.4028	34.0588

Table 5.6: Unsupervised Proxy Metrics Results

Both LDA and BERT demonstrate high Cluster Agreement scores, suggesting their embedding spaces effectively capture underlying structures. BERT achieved the highest Reciprocal Agreement score 0.7716 and the lowest Mutual Rank Score 5.9256, indicating a strong ability to identify mutually relevant pairs with consistent, high-quality reciprocal rankings. In contrast, TF-IDF and Word2Vec scored much lower across all three metrics, underscoring their weaker embedding coherence.

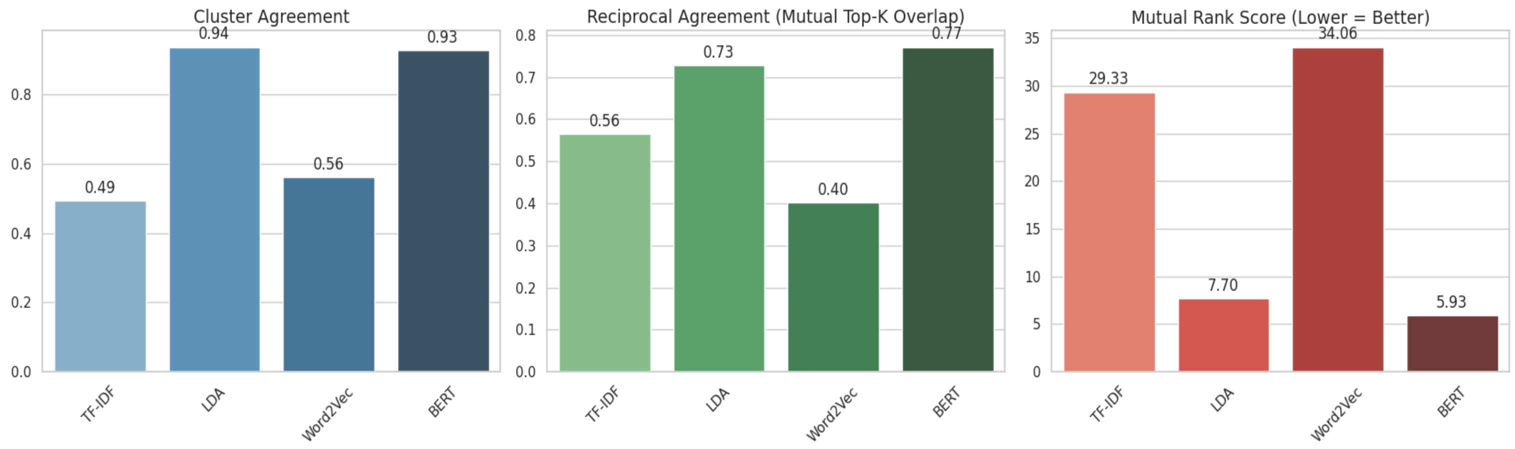


Figure 5.8: Unsupervised Proxy Metrics plot

5.3.3 Visualizing Clusters with PCA

To provide a qualitative understanding of how well each embedding method organizes the resume data, Principal Component Analysis (PCA) was utilized to reduce the dimensionality of the resume embeddings to two components. K-Means clustering was then applied with K=5 to these 2D representations, and the resulting clusters were visualized using scatter plots. Each point in the scatter plots represents a resume, with its color indicating the cluster assigned by K-Means. These visualizations offer a visual assessment of the separability and distinctness of the clusters formed by each embedding method.

The PCA visualization plots, presented in Figure 5.9, reveal clear differences in the clustering patterns produced by the four methods.

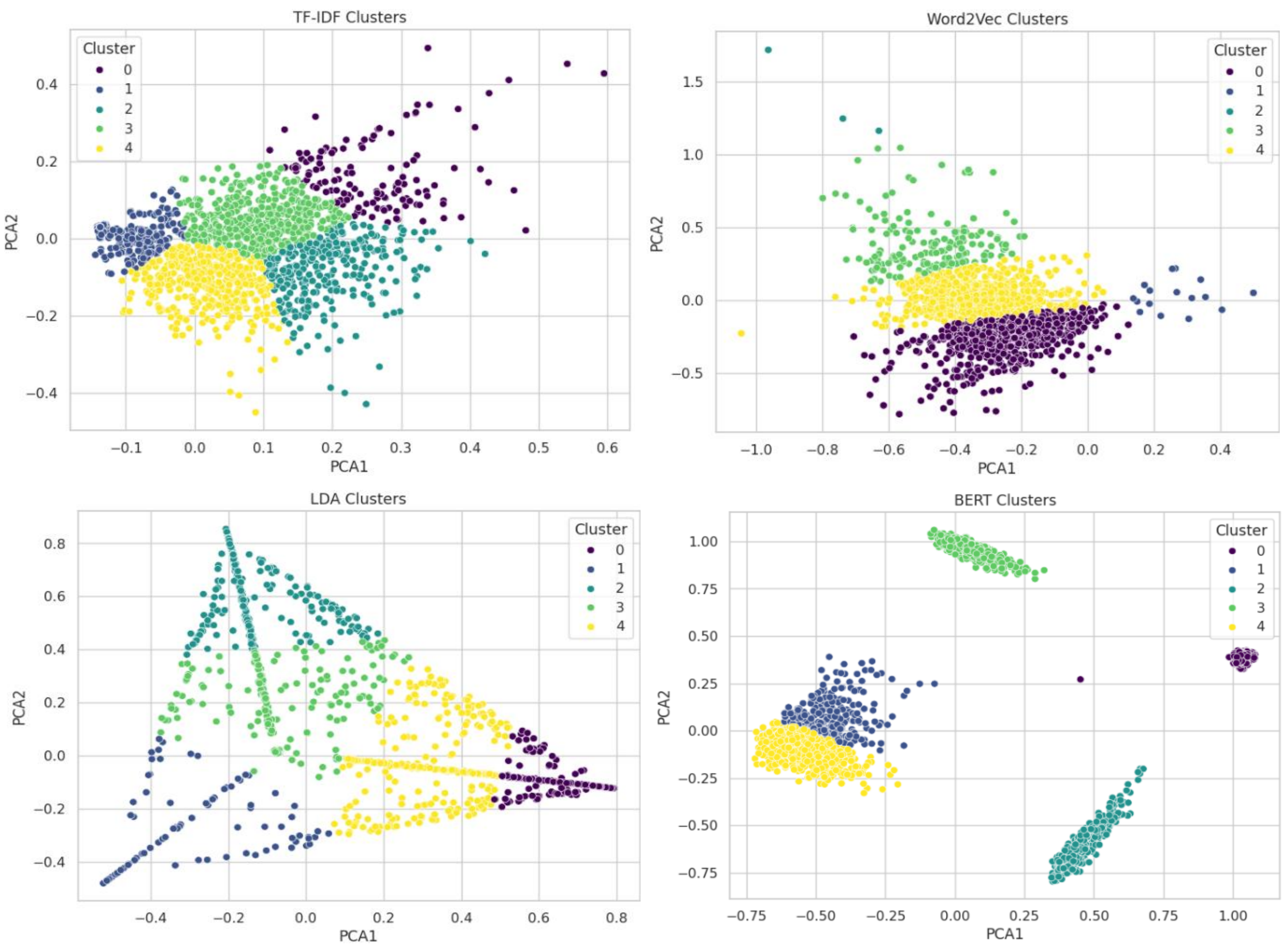


Figure 5.9: PCA Visualization of Resume Clusters for Each Embedding Method

For TF-IDF, the clusters appear somewhat separated in the 2D PCA space, but there is noticeable overlap between several clusters, suggesting that the boundaries between groups are not sharply defined. LDA demonstrates more defined and somewhat triangular-shaped clusters with less overlap compared to TF-IDF. This indicates better separation of the resume data into distinct groups in the reduced dimensionality, aligning with its higher Cluster Agreement score. Word2Vec also shows some degree of separation in its clusters, but similar to TF-IDF, there is noticeable overlap, suggesting less clear distinctions between the groups. BERT produces visually the most distinct and well-separated clusters among all the methods. The groupings are clear with minimal overlap, providing a strong qualitative confirmation of its high Cluster Agreement score and its ability to capture underlying structures that lead to meaningful and separable clusters in the resume data.

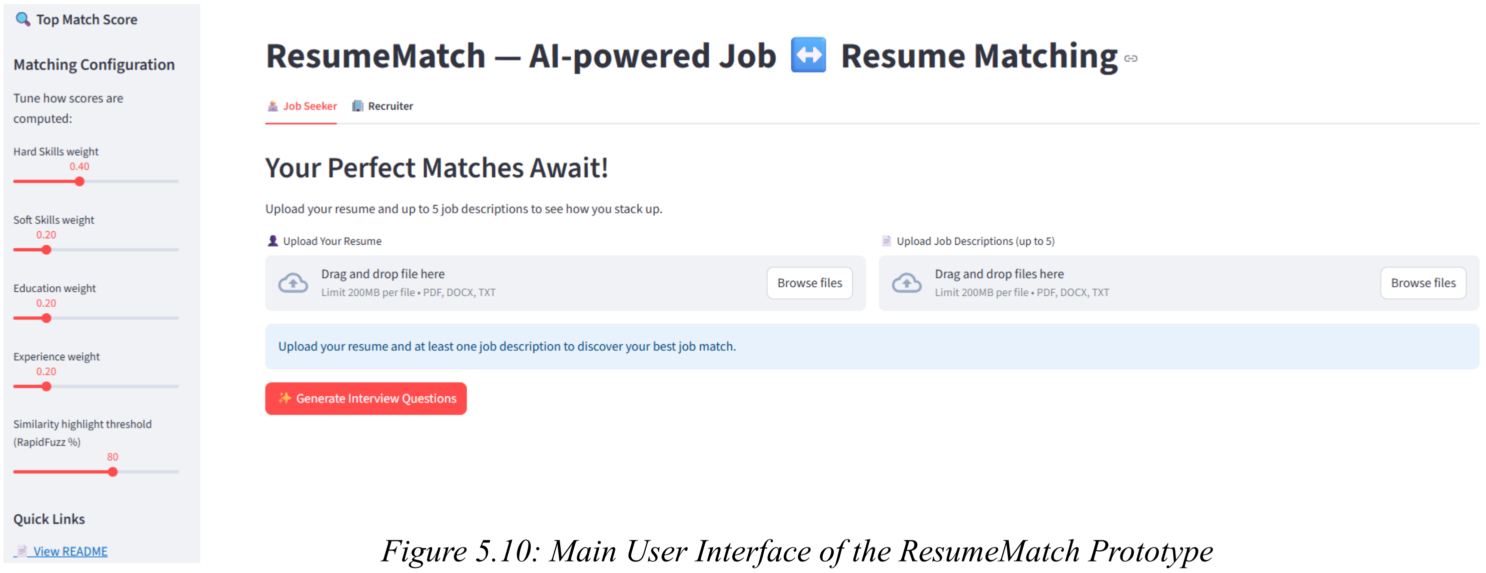
The visual separation of clusters in the PCA plots generally aligns with the quantitative Cluster Agreement metric, providing complementary evidence of how well each embedding method structures the data. BERT and LDA, which performed well in the Cluster Agreement metric, also show more visually separated clusters.

5.3.4 Conclusion on Unsupervised Evaluations (Resume-Job Matching)

Based on the comprehensive unsupervised evaluation, the results indicate that both LDA and BERT embedding methods demonstrate promising performance for the job-resume matching task. LDA consistently excels in the Separation@K metric, showing the strongest ability to differentiate between highly relevant and irrelevant resumes. BERT exhibits strong overall performance, achieving high Cluster Agreement and the highest Reciprocal Agreement with the lowest Mutual Rank Score. Its PCA visualization also showed the most distinct and well-separated clusters. The choice between them depends on the specific priorities of the matching application.

5.4 Prototype UI (Streamlit screenshots)

- **Main UI**
  The figure 5.10 shows the primary user interface of the ResumeMatch application, where users can upload their resume and job description to discover how well they align, and Recruiter can find the perfect candidate. The sidebar on the left allows for adjusting the weighted match score for different criteria like skills, education, and experience and adjusting the threshold of the Rapid/Fuzz match.



Top Match Score

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.40

Soft Skills weight 0.20

Education weight 0.20

Experience weight 0.20

Similarity highlight threshold (RapidFuzz %) 80

Quick Links

View README

ResumeMatch — AI-powered Job ↔ Resume Matching

Job Seeker | Recruiter

Your Perfect Matches Await!

Upload your resume and up to 5 job descriptions to see how you stack up.

Upload Your Resume

Drag and drop file here
Limit 200MB per file • PDF, DOCX, TXT
Browse files

Upload Job Descriptions (up to 5)

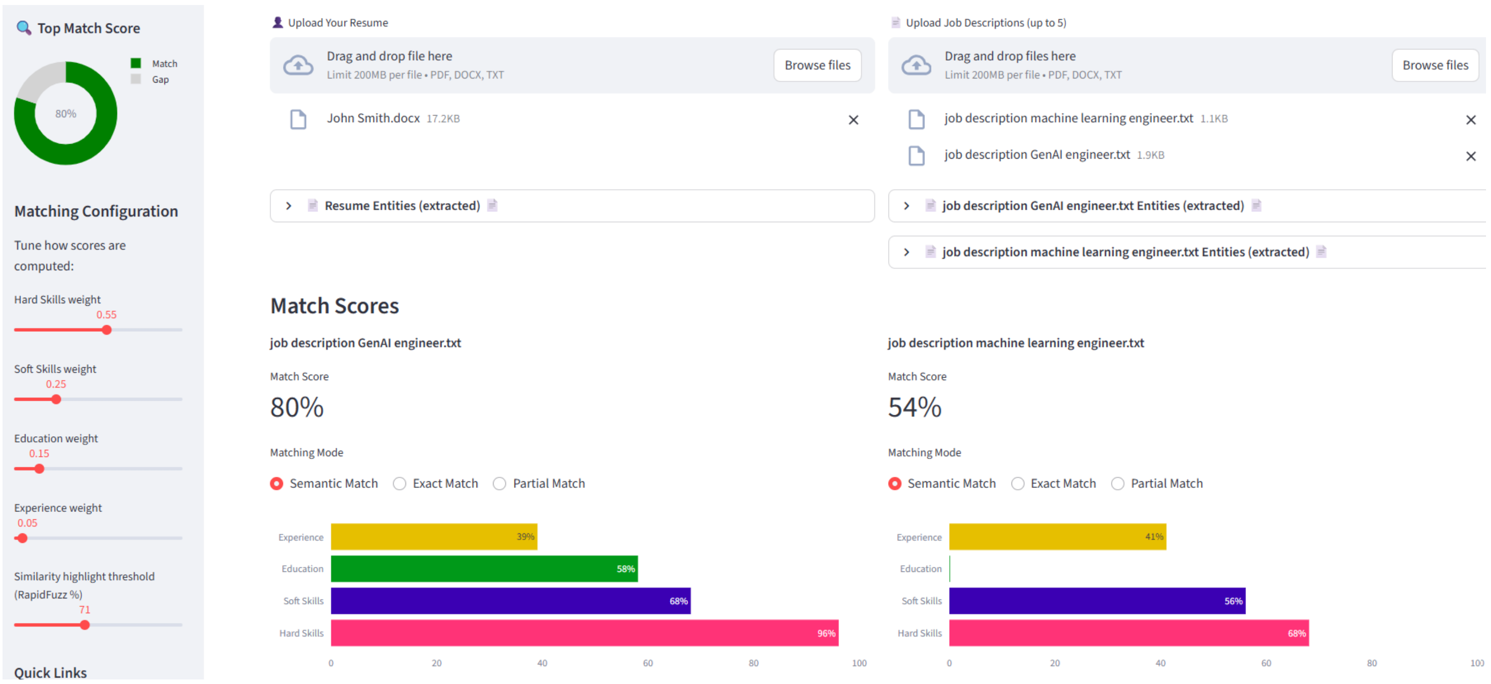
Drag and drop files here
Limit 200MB per file • PDF, DOCX, TXT
Browse files

Upload your resume and at least one job description to discover your best job match.

Generate Interview Questions

Figure 5.10: Main User Interface of the ResumeMatch Prototype

- **Jobseeker tab**
  Figure 5.11 displays the Job Seeker tab after uploading a resume and job description, showing the extracted entities across key sections like skills, education, and experience and final match scores for up to 5 jobs. Users can interact with the Bar Matching Mode selector to switch between semantic, exact, or partial/ fuzzy matching and view dynamic bar charts reflecting their job fit.


🔍 Top Match Score

🟩 Match
▨ Gap

80%

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.55

Soft Skills weight 0.25

Education weight 0.15

Experience weight 0.05

Similarity highlight threshold (RapidFuzz %) 71

Quick Links

Upload Your Resume

Drag and drop file here
Limit 200MB per file • PDF, DOCX, TXT
Browse files

John Smith.docx 17.2KB

Resume Entities (extracted)

Upload Job Descriptions (up to 5)

Drag and drop files here
Limit 200MB per file • PDF, DOCX, TXT
Browse files

job description machine learning engineer.txt 1.1KB
job description GenAI engineer.txt 1.9KB

job description GenAI engineer.txt Entities (extracted)

job description machine learning engineer.txt Entities (extracted)

Match Scores

job description GenAI engineer.txt

Match Score
80%

Matching Mode
[✓ Semantic Match] Exact Match Partial Match

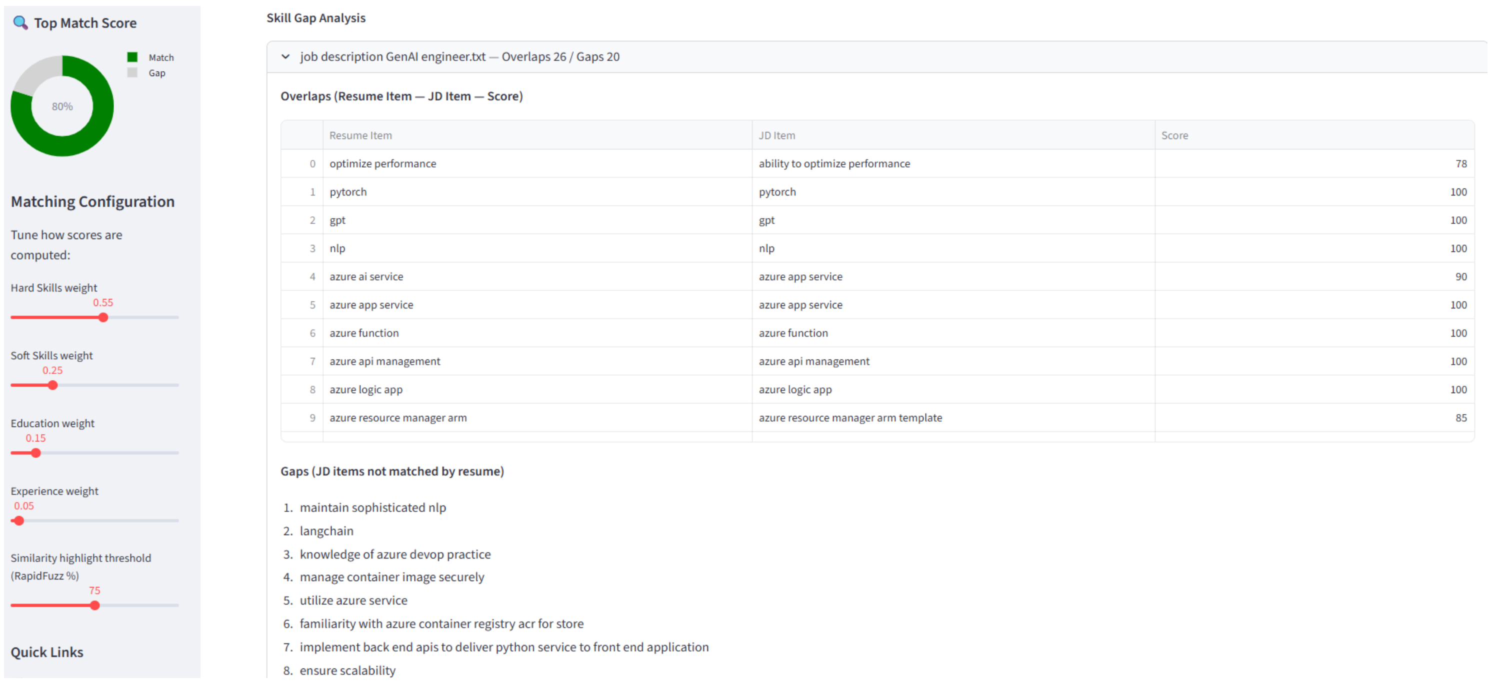
job description machine learning engineer.txt

Match Score
54%

Matching Mode
[✓ Semantic Match] Exact Match Partial Match

Figure 5.11: Job Seeker Dashboard with Matching Results

- **Skill Gap Analysis**
  Figure 5.12 provides a granular breakdown of the skill gap analysis. The visualization helps a job seeker identify which skills they have that are relevant to the job and which are missing, allowing them to optimize their resume.


🔍 Top Match Score

🟩 Match
▨ Gap

80%

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.55

Soft Skills weight 0.25

Education weight 0.15

Experience weight 0.05

Similarity highlight threshold (RapidFuzz %) 75

Quick Links

Skill Gap Analysis

job description GenAI engineer.txt — Overlaps 26 / Gaps 20

Overlaps (Resume Item — JD Item — Score)

	Resume Item	JD Item	Score
0	optimize performance	ability to optimize performance	78
1	pytorch	pytorch	100
2	gpt	gpt	100
3	nlp	nlp	100
4	azure ai service	azure app service	90
5	azure app service	azure app service	100
6	azure function	azure function	100
7	azure api management	azure api management	100
8	azure logic app	azure logic app	100
9	azure resource manager arm	azure resource manager arm template	85

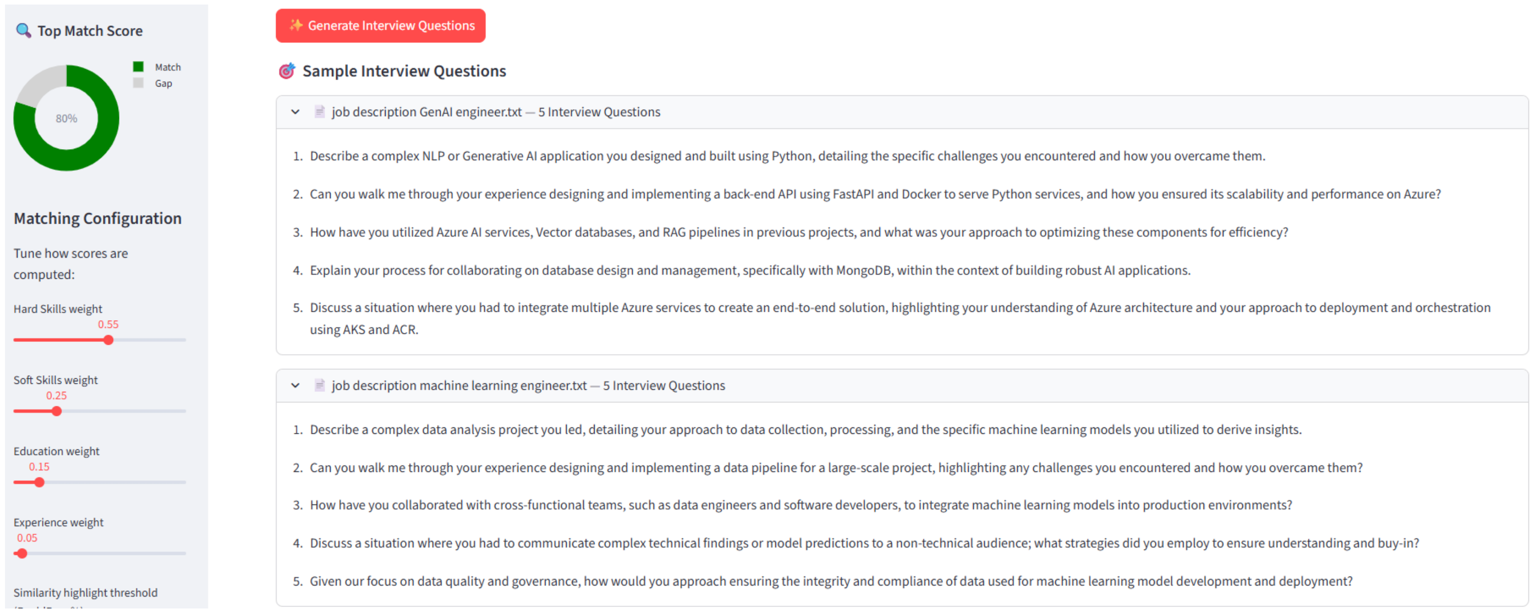
Gaps (JD items not matched by resume)

1. maintain sophisticated nlp
2. langchain
3. knowledge of azure devop practice
4. manage container image securely
5. utilize azure service
6. familiarity with azure container registry acr for store
7. implement back end apis to deliver python service to front end application
8. ensure scalability

Figure 5.12: Detailed Skill Gap Analysis Breakdown

- **Interview Questions**

  Figure 5.13 shows a list of LLM (Google Gemini 2.5 Flash Lite) generated interview questions that are tailored to the content of the uploaded job description. This feature helps job seekers prepare for interviews by focusing on key competencies and experiences



Top Match Score

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.55

Soft Skills weight 0.25

Education weight 0.15

Experience weight 0.05

Similarity highlight threshold

Generate Interview Questions

Sample Interview Questions

job description GenAI engineer.txt — 5 Interview Questions

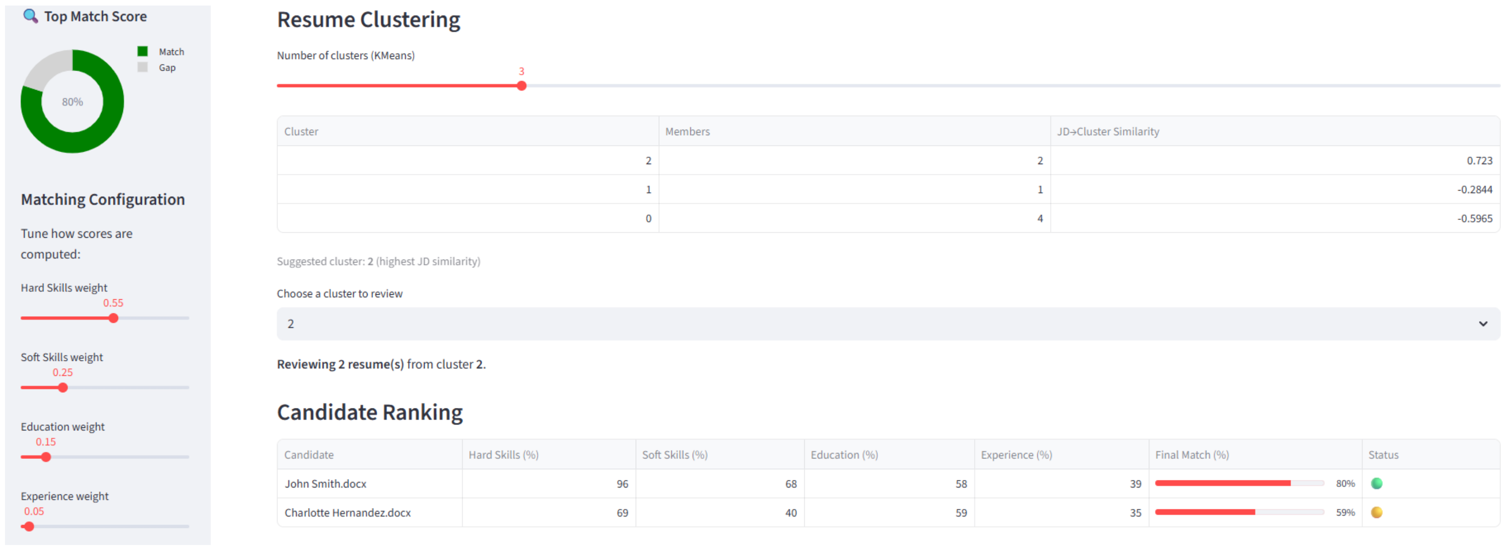
1. Describe a complex NLP or Generative AI application you designed and built using Python, detailing the specific challenges you encountered and how you overcame them.
2. Can you walk me through your experience designing and implementing a back-end API using FastAPI and Docker to serve Python services, and how you ensured its scalability and performance on Azure?
3. How have you utilized Azure AI services, Vector databases, and RAG pipelines in previous projects, and what was your approach to optimizing these components for efficiency?
4. Explain your process for collaborating on database design and management, specifically with MongoDB, within the context of building robust AI applications.
5. Discuss a situation where you had to integrate multiple Azure services to create an end-to-end solution, highlighting your understanding of Azure architecture and your approach to deployment and orchestration using AKS and ACR.

job description machine learning engineer.txt — 5 Interview Questions

1. Describe a complex data analysis project you led, detailing your approach to data collection, processing, and the specific machine learning models you utilized to derive insights.
2. Can you walk me through your experience designing and implementing a data pipeline for a large-scale project, highlighting any challenges you encountered and how you overcame them?
3. How have you collaborated with cross-functional teams, such as data engineers and software developers, to integrate machine learning models into production environments?
4. Discuss a situation where you had to communicate complex technical findings or model predictions to a non-technical audience; what strategies did you employ to ensure understanding and buy-in?
5. Given our focus on data quality and governance, how would you approach ensuring the integrity and compliance of data used for machine learning model development and deployment?

Figure 5.13: LLM-Generated Interview Questions

- **Recruiter Tab**



Top Match Score

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.55

Soft Skills weight 0.25

Education weight 0.15

Experience weight 0.05

Resume Clustering

Number of clusters (KMeans) 3

Cluster	Members	JD→Cluster Similarity
2	2	0.723
1	1	-0.2844
0	4	-0.5965

Suggested cluster: 2 (highest JD similarity)

Choose a cluster to review

2

Reviewing 2 resume(s) from cluster **2**.

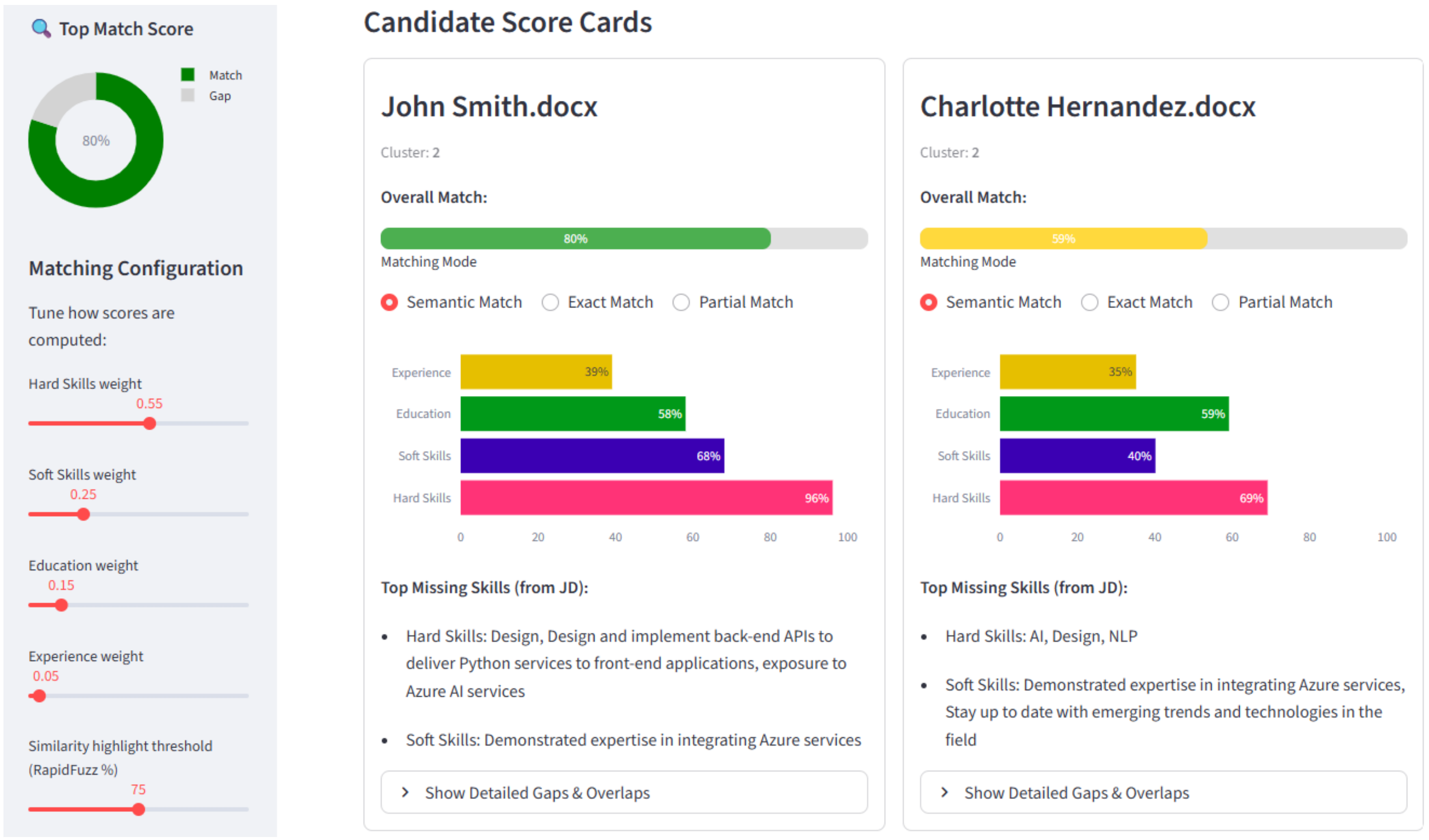
Candidate Ranking

Candidate	Hard Skills (%)	Soft Skills (%)	Education (%)	Experience (%)	Final Match (%)	Status
John Smith.docx	96	68	58	39	80%	🟢
Charlotte Hernandez.docx	69	40	59	35	59%	🟡

Figure 5.14: Resume Clustering and candidate Ranking

Figure 5.14 shows the Recruiter tab, which provides a dashboard for filtering and ranking candidates based on a single job description. The interface allows recruiters to efficiently manage and evaluate a large number of resumes using clustering, compute the similarity score between job description and each cluster centroid. It filters out the clusters of the relevant candidates. By automating resume clustering and job description matching, the system significantly reduces the time required to screen resumes manually.

- **Candidate Score Cards**
  Figure 5.15 displays each candidate score card which contains final weightage match score, top 3 missing skills, and detailed skills gap analysis, helping recruiters quickly identify top fits and streamline the selection process.



Top Match Score

Matching Configuration

Tune how scores are computed:

Hard Skills weight 0.55

Soft Skills weight 0.25

Education weight 0.15

Experience weight 0.05

Similarity highlight threshold (RapidFuzz %) 75

Candidate Score Cards

John Smith.docx

Cluster: 2

Overall Match: 80%

Matching Mode

[✓ Semantic Match] Exact Match Partial Match

Top Missing Skills (from JD):

- Hard Skills: Design, Design and implement back-end APIs to deliver Python services to front-end applications, exposure to Azure AI services
- Soft Skills: Demonstrated expertise in integrating Azure services

> Show Detailed Gaps & Overlaps

Charlotte Hernandez.docx

Cluster: 2

Overall Match: 59%

Matching Mode

[✓ Semantic Match] Exact Match Partial Match

Top Missing Skills (from JD):

- Hard Skills: AI, Design, NLP
- Soft Skills: Demonstrated expertise in integrating Azure services, Stay up to date with emerging trends and technologies in the field

> Show Detailed Gaps & Overlaps

Figure 5.15: Candidate Score Cards

6. DISCUSSION, CONCLUSION, AND FUTURE WORK

Discussion

The experimental findings reveal a clear hierarchy of effectiveness for the methods employed in skill extraction and resume-job matching. For skill extraction from job descriptions, the Custom NER model and the LLM-based approach (Google Gemini 2.5 Flash Lite) demonstrated superior performance over all rule-based and general-purpose embedding methods. The Custom NER model's exceptional recall and F1-score are directly attributable to its fine-tuning on a domain-specific dataset. A notable improvement in performance was observed after expanding the training data from 50 to 150 manually annotated samples, underscoring the critical role of data quantity and quality in achieving high accuracy. The LLM, while a close competitor, showcased its strength in generating structured output and understanding context, particularly for extracting education entities, despite its operational limitations.

For resume-job matching, the unsupervised evaluation highlighted the distinct strengths of Latent Dirichlet Allocation (LDA) and BERT. LDA excelled at ranking, with the highest scores on the Separation@K metric, making it highly effective for differentiating between the most and least relevant candidates. Conversely, BERT proved to be a more nuanced relevance engine, demonstrating superior performance in identifying mutually relevant pairs and producing visually distinct clusters in PCA space. The baseline methods, TF-IDF and Word2Vec, generally failed to capture these semantic relationships effectively.

Implications

The results of this study have significant implications for the development of AI-driven recruitment platforms. The superior performance of the Custom NER model suggests that for high-precision, domain-specific tasks like information extraction, a fine-tuned model can still outperform powerful, general-purpose LLMs. This highlights a strategic trade-off between the flexibility of LLMs and the accuracy of specialized models. Furthermore, the complementary strengths of LDA and BERT in the matching task indicate that a single embedding method may not be optimal for all objectives. An effective system could leverage a hybrid approach, using LDA for initial candidate filtering and BERT for a final, high-fidelity ranking. This modular design offers a path to creating more robust and purpose-built matching systems.

Limitations

Several limitations were encountered during this research. The Custom NER model, despite its high performance on job descriptions, struggled with resume parsing. Specifically, it was unable to consistently extract experience entries that included dates, as this structured information was not present in its job description training data. Additionally, due to time constraints, the 100-sample test set used for model evaluation was annotated with the assistance of a large language model (ChatGPT 4.0). While this expedited the evaluation, it introduces potential sources of bias and imprecision compared to a fully manual annotation process. A more reliable and definitive assessment of model performance would require a manually annotated test set, thereby establishing a stronger ground truth for validation.

Future Work

Future work should focus on addressing the identified limitations and exploring new avenues for improvement. Key areas include a supervised evaluation with a manually annotated test set, the development of a hybrid matching model to combine the strengths of LDA and BERT, and the fine-tuning of a Custom NER model on a larger, manually annotated resume corpus to improve the robustness of experience and education section extraction. Additionally, a crucial next step is to explore a conversational chat feature to provide personalized feedback and suggestions for resume improvement based on a skill gap analysis, enhancing the user experience. The system could also be expanded to support multilingual functionality, broadening its applicability to a global user base. Incorporating explainable AI (XAI) techniques and expanding the model to include quantitative attributes like years of experience could further enhance the system's accuracy and user trust.

Conclusion

This research introduces ResumeMatch, a dual-role platform that integrates a custom, transformer-based NER built on spaCy framework for precise entity extraction with embedding-driven semantic matching. By fine-tuning on domain-specific data and comparing multiple embedding strategies, we achieved robust performance across parsing and matching tasks. Our findings underscore the value of specialized models for high-precision extraction and the importance of choosing embedding techniques according to matching goals. ResumeMatch provides a transparent, scalable framework for both job seekers and recruiters, laying a solid foundation for further advancements in AI-assisted hiring.

7. REFERENCES

1. Ayishathahira, C. H., Sreejith, C., & Raseek, C. (2018). Combination of Neural Networks and Conditional Random Fields for Efficient Resume Parsing. *2018 International CET Conference on Control, Communication, and Computing (IC4)*, 388–393. https://doi.org/10.1109/CETIC4.2018.8530883

2. Bevara, R. V. K., Mannuru, N. R., Karedla, S. P., Lund, B., Xiao, T., Pasem, H., Dronavalli, S. C., & Rupeshkumar, S. (2025). Resume2Vec: Transforming Applicant Tracking Systems with Intelligent Resume Embeddings for Precise Candidate Matching. *Electronics (Basel)*, *14*(4), 794–. https://doi.org/10.3390/electronics14040794

3. Bhatt, A., Mittal, S., Uniyal, A., Tiwari, P., Jyala, D., & Singh, D. (2024). Resume Analyzer based on MapReduce and Machine Learning. *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, *2*, 1–5. https://doi.org/10.1109/IATMSI60426.2024.10503467

4. Burrell, D. N., & Mcandrew, I. (2023). Exploring the Ethical Dynamics of the Use of Artificial Intelligence (AI) in Hiring in Healthcare Organizations. *Land Forces Academy Review*, *28*(4), 309–321. https://doi.org/10.2478/raft-2023-0037

5. Delecraz, S., Eltarr, L., Becuwe, M., Bouxin, H., Boutin, N., & Oullier, O. (2022). Making recruitment more inclusive: unfairness monitoring with a job matching machine-learning algorithm. *2022 IEEE/ACM International Workshop on Equitable Data & Technology (FairWare)*, 34–41. https://doi.org/10.1145/3524491.3527309

6. Fettach, Y., Bahaj, A., & Ghogho, M. (2025). Skill Demand Forecasting Using Temporal Knowledge Graph Embeddings. https://doi.org/10.48550/arxiv.2504.07233

7. Frissen, R., Adebayo, K. J., & Nanda, R. (2023). A machine learning approach to recognize bias and discrimination in job advertisements. *AI & Society*, *38*(2), 1025–1038. https://doi.org/10.1007/s00146-022-01574-0

8. Gaur, B., Saluja, G. S., Sivakumar, H. B., & Singh, S. (2021). Semi-supervised deep learning based named entity recognition model to parse education section of resumes. *Neural Computing & Applications*, *33*(11), 5705–5718. https://doi.org/10.1007/s00521-020-05351-2

9. J, P. V., P, S. N. J., Gopinath, S., S, U., & C.R., K. (2024). Resume Analyzer and Skill Enhancement Recommender System. *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, 1–6. https://doi.org/10.1109/APCIT62007.2024.10673530

10. Kinger, S., Kinger, D., Thakkar, S., & Bhake, D. (2024). Towards smarter hiring: resume parsing and ranking with YOLOv5 and DistilBERT. *Multimedia Tools and Applications*, *83*(35), 82069–82087. https://doi.org/10.1109/s11042-024-18778-9

11. Özcan, G., & Ögüdücü, S. G. (2017). Applying Classifications Techniques in Job Recommendation System for Matching of Candidates and Advertisements. *International Journal of Intelligent Computing Research*, *8*(1), 798–806. https://doi.org/10.20533/ijicr.2042.4655.2017.0098

12. Pabalkar, S., Patel, P., Choudhary, R., Panoch, V., Yadav, S., & Ghogale, H. (2024). Resume Analyzer Using Natural Language Processing (NLP). *2024 International Conference on Intelligent Systems and Advanced Applications (ICISAA)*, 1–6. https://doi.org/10.1109/ICISAA62385.2024.10828940

13. Singh, S., & Gupta, P. (2023). Talent Recommendation Engine with Real-Time Feedback Loop. *Proceedings of the Third International Conference on AI-ML Systems*, 1–4. https://doi.org/10.1145/3639856.3639911

14. Sruthi, P., Adithya, P. N. V. K. G., Suleman, M. D., Kunal, P., & Gairola, S. P. (2023). Smart Resume Analyser: A Case Study using RNN-based Keyword Extraction. *E3S Web of Conferences*, *430*, 1023–. https://doi.org/10.1051/e3sconf/202343001023

15. Zhang, L., & Yencha, C. (2022). Examining perceptions towards hiring algorithms. *Technology in Society*, *68*, 101848–.https://doi.org/10.1016/j.techsoc.2021.101848

16. Zhao, J., Wang, J., Sigdel, M., Zhang, B., Hoang, P., Liu, M., & Korayem, M. (2021). Embedding-based Recommender System for Job to Candidate Matching on Scale. https://doi.org/10.48550/arxiv.2107.00221

8. APPENDIX – A | DATASET & GITHUB LINK

Dataset

1. Resume Dataset (Kaggle):
   - Contains 2485 resumes.
   - Source: Resume Dataset
2. Job Postings Dataset (Kaggle):

- Includes 123,849 records of job postings
- Source: Job Postings Dataset

GitHub Repository link:
https://github.com/rabiadanish/ResumeMatch

9. APPENDIX – B | LLM PROMPTS

LLM Prompt for job description parsing

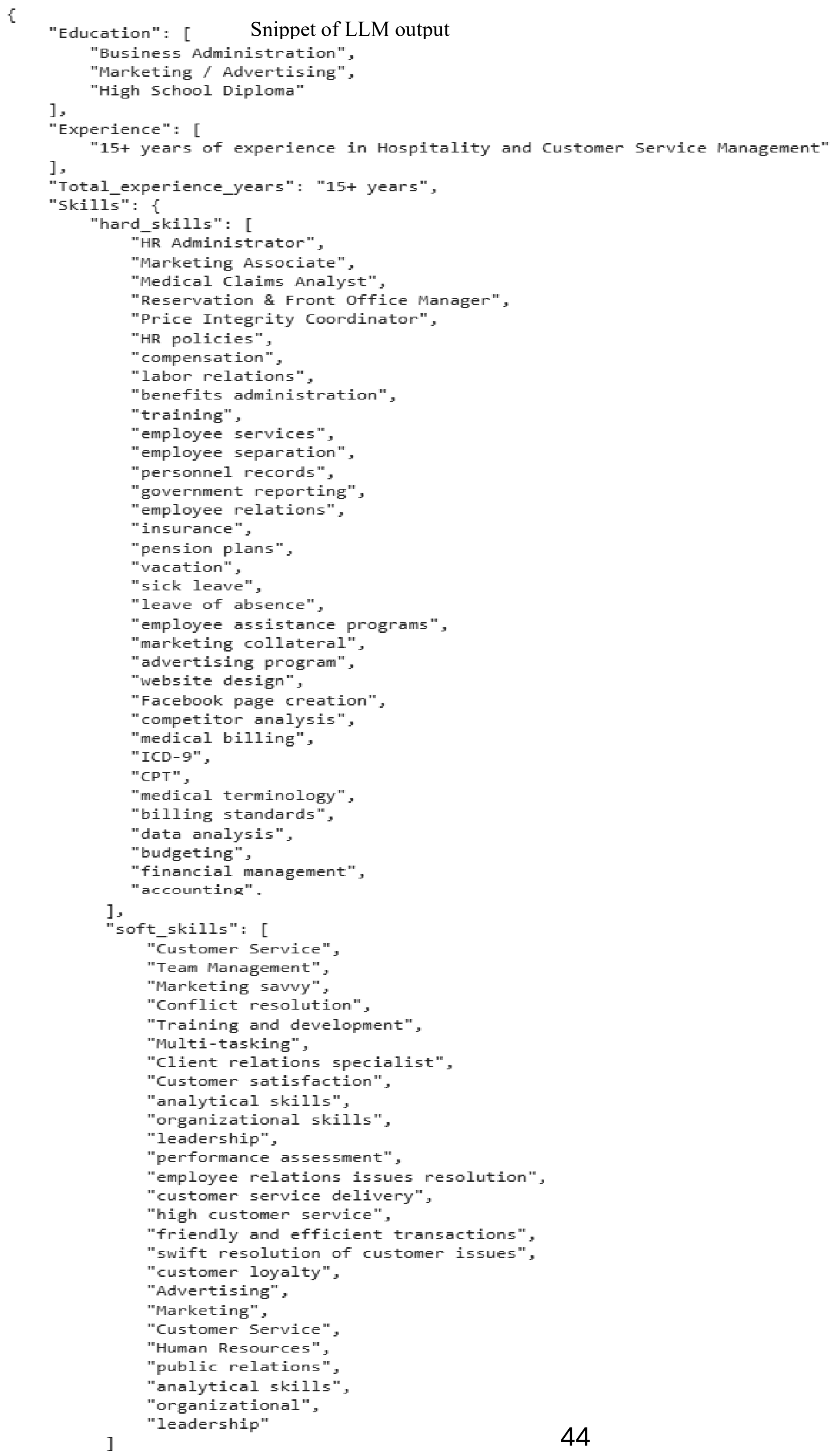
```
You are an expert AI trained to extract structured information from job descriptions.
Please analyze the following job description and extract all unique key terms or phrases into four categories:

- **hard_skills**: Concrete, technical, or domain-specific skills or tools (e.g., Python, AutoCAD, SQL, financial modeling) **that are explicitly stated**.
- **soft_skills**: Clearly mentioned interpersonal or behavioral traits (e.g., teamwork, time management, leadership).
- **education**: Exact degrees, certifications, or academic requirements mentioned in the job description (e.g., "Bachelor's in Engineering", "MBA", "Certified Public Accountant").
- **experience**: Specific requirements around years of experience, industry experience, or role-based experience mentioned in the job text (e.g., "5+ years in project management", "experience in healthcare").

Only return values that are **verbatim or clearly paraphrased from the text**. Do not include generic skills, vague summaries, or industry norms unless they are stated directly in the job description.
Do not invent, hallucinate, or return generalized content.
Respond ONLY in this JSON format (no explanations or extra text):
{{
  "hard_skills": [...],
  "soft_skills": [...],
  "education": [...],
  "experience": [...]
}}
```

LLM Prompt for Resume parsing

```
You are an expert resume parser. Read the resume text below and extract structured information in JSON format with the following fields:
```

```
1. Education: [Exact degrees, certifications, or academic requirements mentioned in the job description, e.g., "Bachelor's in Computer Science", "Master's in Business Administration",
            (e.g., "Bachelor’s in Engineering", "MBA", "Certified Public Accountant")
]
2. Experience: [Specific requirements around years of experience, industry experience, or role-based experience mentioned in the job text (e.g., "5+ years in project management", "experience in healthcare")
]
3. Total_experience_years: [total years of professional experience, e.g. 5+ years]
4. Skills: {{
   "hard_skills": [Should Extraxt All hard/technical skills mentioned in the resume and job responsibilities],
   "soft_skills": [Should extract All soft/interpersonal/communication/managerial skills mentioned or implied in the resume and job responsibilities]
}}
Only return a clean JSON dictionary with the above fields.
Exclude any field if the information is missing or unclear.
Use double quotes around all field values.
Do not include generic skills, vague summaries, or industry norms unless they are stated directly in the Resume.
```

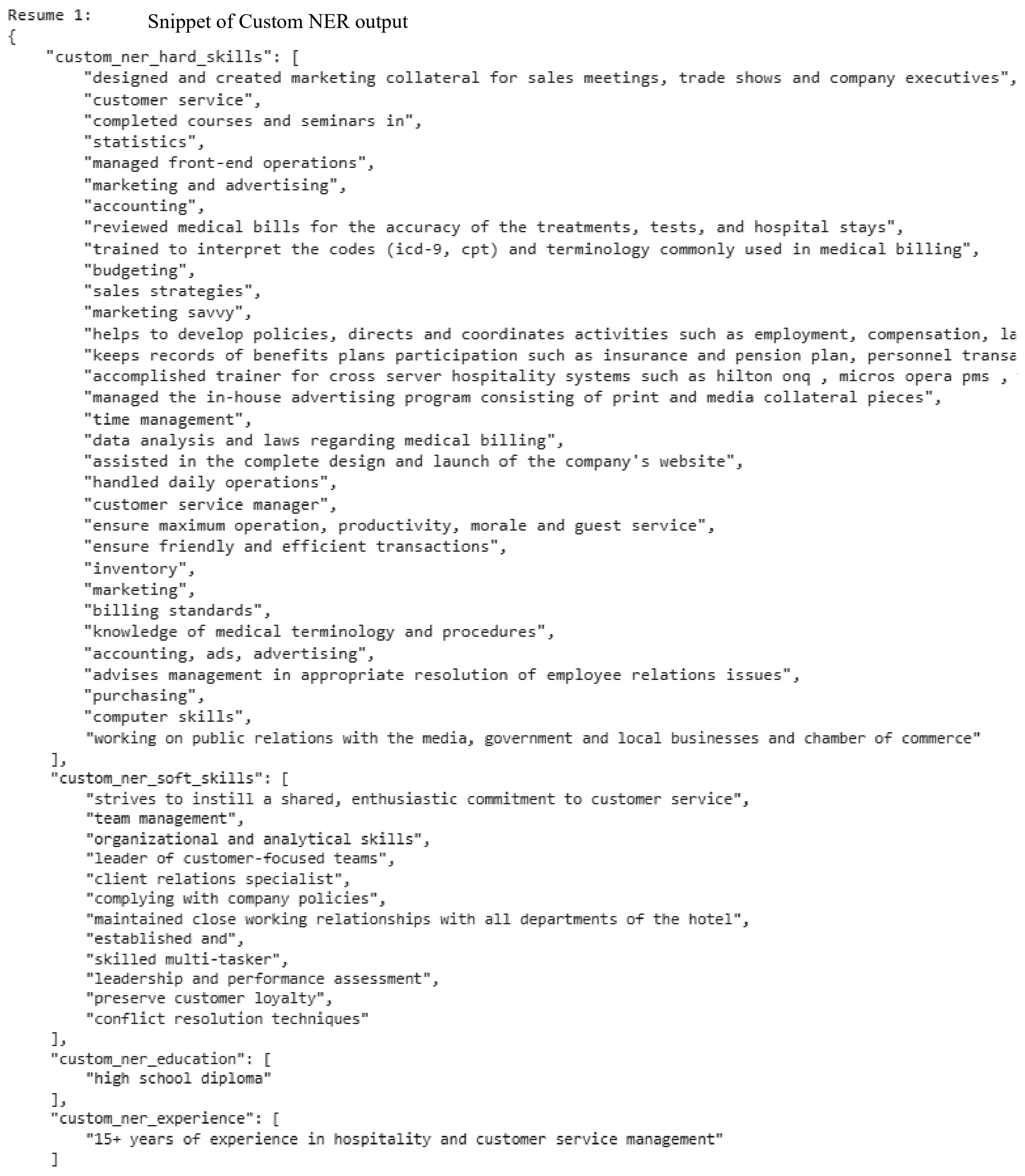
10. APPENDIX – C | RESUME PARSING OUTPUT



Snippet of LLM output

```
{
    "Education": [
        "Business Administration",
        "Marketing / Advertising",
        "High School Diploma"
    ],
    "Experience": [
        "15+ years of experience in Hospitality and Customer Service Management"
    ],
    "Total_experience_years": "15+ years",
    "Skills": {
        "hard_skills": [
            "HR Administrator",
            "Marketing Associate",
            "Medical Claims Analyst",
            "Reservation & Front Office Manager",
            "Price Integrity Coordinator",
            "HR policies",
            "compensation",
            "labor relations",
            "benefits administration",
            "training",
            "employee services",
            "employee separation",
            "personnel records",
            "government reporting",
            "employee relations",
            "insurance",
            "pension plans",
            "vacation",
            "sick leave",
            "leave of absence",
            "employee assistance programs",
            "marketing collateral",
            "advertising program",
            "website design",
            "Facebook page creation",
            "competitor analysis",
            "medical billing",
            "ICD-9",
            "CPT",
            "medical terminology",
            "billing standards",
            "data analysis",
            "budgeting",
            "financial management",
            "accounting",
         ],
         "soft_skills": [
             "Customer Service",
             "Team Management",
             "Marketing savvy",
             "Conflict resolution",
             "Training and development",
             "Multi-tasking",
             "Client relations specialist",
             "Customer satisfaction",
             "analytical skills",
             "organizational skills",
             "leadership",
             "performance assessment",
             "employee relations issues resolution",
             "customer service delivery",
             "high customer service",
             "friendly and efficient transactions",
             "swift resolution of customer issues",
             "customer loyalty",
             "Advertising",
             "Marketing",
             "Customer Service",
             "Human Resources",
             "public relations",
             "analytical skills",
             "organizational",
             "leadership"
         ]
```

Snippet of Custom NER output



```
Resume 1:
{
    "custom_ner_hard_skills": [
        "designed and created marketing collateral for sales meetings, trade shows and company executives",
        "customer service",
        "completed courses and seminars in",
        "statistics",
        "managed front-end operations",
        "marketing and advertising",
        "accounting",
        "reviewed medical bills for the accuracy of the treatments, tests, and hospital stays",
        "trained to interpret the codes (icd-9, cpt) and terminology commonly used in medical billing",
        "budgeting",
        "sales strategies",
        "marketing savvy",
        "helps to develop policies, directs and coordinates activities such as employment, compensation, la
        "keeps records of benefits plans participation such as insurance and pension plan, personnel transa
        "accomplished trainer for cross server hospitality systems such as hilton onq , micros opera pms ,
        "managed the in-house advertising program consisting of print and media collateral pieces",
        "time management",
        "data analysis and laws regarding medical billing",
        "assisted in the complete design and launch of the company's website",
        "handled daily operations",
        "customer service manager",
        "ensure maximum operation, productivity, morale and guest service",
        "ensure friendly and efficient transactions",
        "inventory",
        "marketing",
        "billing standards",
        "knowledge of medical terminology and procedures",
        "accounting, ads, advertising",
        "advises management in appropriate resolution of employee relations issues",
        "purchasing",
        "computer skills",
        "working on public relations with the media, government and local businesses and chamber of commerce"
    ],
    "custom_ner_soft_skills": [
        "strives to instill a shared, enthusiastic commitment to customer service",
        "team management",
        "organizational and analytical skills",
        "leader of customer-focused teams",
        "client relations specialist",
        "complying with company policies",
        "maintained close working relationships with all departments of the hotel",
        "established and",
        "skilled multi-tasker",
        "leadership and performance assessment",
        "preserve customer loyalty",
        "conflict resolution techniques"
    ],
    "custom_ner_education": [
        "high school diploma"
    ],
    "custom_ner_experience": [
        "15+ years of experience in hospitality and customer service management"
    ]
```