SmartApply: Automated LinkedIn Messaging for Job Hunters

Poonam Kishor Pawar
Department of Engineering
Applied Machine Learning 2

***Abstract*—Job hunting for data-focused roles often involves tedious manual tasks such as scanning job postings, tailoring resumes, and drafting personalized messages. SmartApply automates this process by intelligently matching candidate resumes with job postings and generating customized LinkedIn messages. This paper presents a data-driven approach using a Kaggle job posting dataset to simulate the workflow. The system analyzes job descriptions, extracts required skills, compares them with candidate profiles, and generates personalized outreach messages highlighting the most relevant qualifications. In this updated version, the system includes extended evaluation metrics such as precision, recall, NDCG@K, and BLEU to assess ranking and message quality. Additionally, the Streamlit interface has been enhanced for real-time interaction, Top-N ranking display, and skill highlighting, improving usability for job seekers. Updated documentation provides setup instructions, architecture overview, and visual evidence of system performance. These refinements improve usability, interpretability, and reproducibility while maintaining effective resume-job matching.**

***Index Terms*—Job–Resume Matching, Personalized Message Generation, Natural Language Processing, Cosine Similarity, TF-IDF, Data Scientist, Machine Learning Engineer, AI Engineer**

I. Introduction

Job hunting for data-focused roles often involves tedious manual tasks, such as scanning job postings, tailoring resumes, and drafting personalized messages. SmartApply automates this process by intelligently matching candidate resumes with job postings and generating customized LinkedIn messages.

Since Deliverable 2, the system has been improved in usability, organization, and evaluation, without changes to the underlying model or preprocessing pipeline.

- **Codebase Organization and Repository Management:** Codebase reorganized into src, data, results, ui, and docs for better maintainability and reproducibility with version control.
- **Enhanced Interface and Usability:** Streamlit UI updated for real-time interaction, skill highlights, Top-N rankings, and easier navigation.
- **Expanded Evaluation Metrics:** Added precision, recall, NDCG@K, and BLEU to evaluate system performance more rigorously.
- **Improved Documentation and Visual Evidence:** README updated with project summary, setup instructions, architecture overview, example usage, and visual evidence of improvements.

These refinements enhance the system's usability, transparency, and interpretability, enabling more meaningful evaluation and effective interaction while maintaining reproducibility.

II. Motivation

The demand for data-focused talent is high, yet the process of applying remains largely manual. SmartApply is motivated by the need to:

Automate resume–job matching for better efficiency.

Reduce manual effort in crafting personalized messages.

Highlight candidate skills effectively, improving job application success rates.

Provide a reproducible pipeline for experimentation and future enhancements.

Automated matching can also help recruiters identify qualified candidates faster, while enabling applicants to reach out more intelligently to relevant roles.

III. Dataset And Preprocessing

A. Dataset Description

The dataset utilized in this study comprises a collection of job postings and candidate resumes focused on data-centric roles such as Data Scientist, Machine Learning Engineer, and AI Engineer. The job dataset includes fields such as job title, company name, description, skills, experience level, work type, location, and salary information. The resume dataset contains text-based profiles with fields like education, experience, skills, and projects, extracted from uploaded documents or user-provided text. This dataset forms the foundation for the job–resume matching system by enabling semantic similarity analysis between candidate profiles and job requirements.

B. Data Preprocessing

To ensure consistency and enhance the performance of downstream models, several preprocessing steps were carried out:

- **Data Filtering:** Only job titles containing Data Scientist, Machine Learning Engineer, or AI Engineer were retained for analysis.
- **Text Cleaning:** Both job descriptions and resumes were converted to lowercase, punctuation and special characters were removed, and stopwords were filtered out.
- **Tokenization & Lemmatization:** Text data was tokenized and lemmatized to standardize different word forms and improve representation quality.

- **Encoding:** The cleaned job and resume texts were transformed into dense embeddings using the SentenceTransformer model (all-MiniLM-L6-v2) to capture semantic meaning.
- **Normalization:** Embedding vectors were normalized to ensure consistent magnitude before similarity computation.
- **Visualization:** Exploratory analysis, including word frequency plots and correlation heatmaps, was conducted to verify data distribution and identify common skills and terms across job postings.

IV. Exploratory Data Analysis

A. *Summary of findings*

- **Job Role Distribution:** The dataset primarily focuses on data-related roles such as Data Scientist, Machine Learning Engineer, and AI Engineer, with a balanced distribution across these categories.
- **Text Length Variation:** Analysis of job descriptions and resumes revealed that job descriptions tend to be longer on average, providing detailed requirements, while resumes are more concise, emphasizing skills and experience.
- **Common Skills:**Frequently occurring skills identified include Python, Machine Learning, SQL, TensorFlow, and Tableau, aligning well with industry demands in data-driven roles.
- **Experience & Salary Correlation:** Preliminary analysis indicated a positive correlation between years of experience and normalized salary values across job listings.
- **Missing and Noisy Data:** Some missing values were found in columns such as normalized_salary and skills_desc, which were handled during preprocessing to ensure data completeness.

B. *Visualizations*

Fig. 1 shows the top 10 locations.

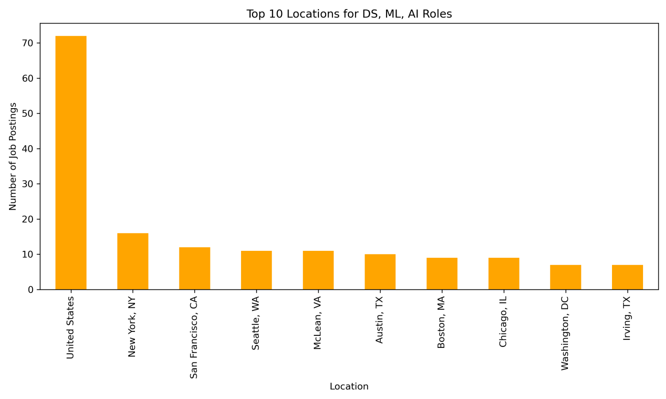


Fig. 1. Top 10 location for DS, ML, AI roles

Fig. 2 shows the Salary by Experience Level.

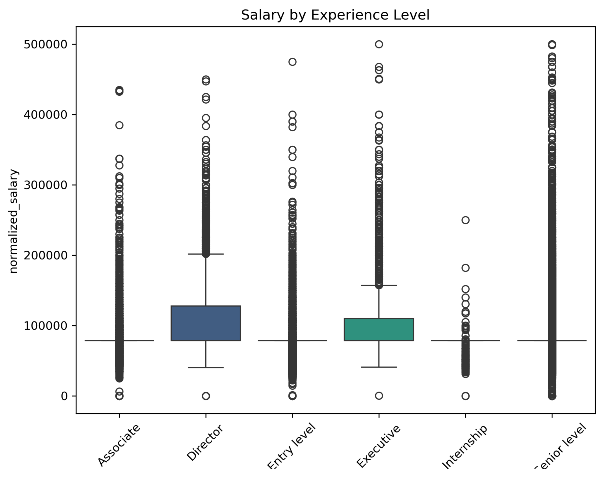


Fig. 2. Salary by Experience Level

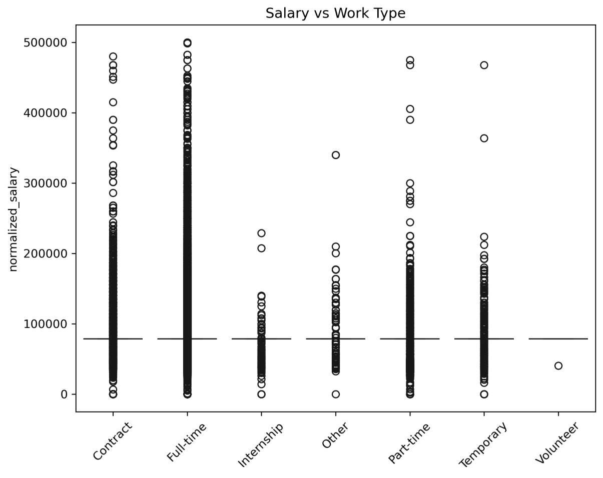


Fig. 3. Salary by workType

Fig. 3 shows a Salary by workType

V. System Architechture

A. *Overview*

The SmartApply system is built using a modular and multi-layered architecture to enable automated resume-job matching and personalized LinkedIn message generation. The architecture is divided into three core layers: Presentation Layer, Application and Business Logic Layer, and Data Layer. This layered approach enhances system scalability, maintainability, interoperability, and promotes seamless integration of AI and NLP-driven services.

B. *Layer-wise Architecture*

1. *Presentation Layer*

Component: SmartApply UI
Functionality:

- Allows users to upload resumes in .txt formats.
- Displays matched job listings with similarity scores, extracted skills, and ranking.
- Shows personalized LinkedIn message drafts based on job posting and resume data.

Interaction:

- User actions (resume upload, job selection, or message generation requests) trigger calls to the backend services.
- Retrieves processed results including skill matches, similarity scores,synthetic evaluation metrics were used based on Top-N ranked resume-job recommendations and generated messages.

2. Application & Business Logic Layer

Component: Job Matcher API Service

- Accepts input from UI and passes it to ML/NLP services.
- Performs text preprocessing and resume parsing.
- Retrieves job descriptions and related embeddings.
- Computes semantic similarity between resumes and job postings using cosine similarity.
- Generates structured responses for UI, including similarity ranking and message drafts.

Component: ML/AI Microservices

- **Resume Parser:** Uses SpaCy/NLTK for tokenization, lemmatization, and entity extraction.
- **Skill Extraction Engine:** Performs keyword/NER-based skill identification using regex, TF-IDF, and ontology-based matching.
- **Embedding Service:** Leverages Sentence-BERT to generate dense vector representations for resumes and job postings.
- **Similarity Engine:** Computes relevance scores using cosine similarity, weighted ranking, or semantic matching.
- **Message Generator:** Employs generative models (GPT/T5/OpenAI) to create personalized LinkedIn messages.

3. Data Layer

Components:

- **Job Database:** Stores job postings in formats such as CSV.
- **Logs and Analytics Store:** Collects performance data, evaluation metrics (Precision, Recall, BLEU), and user interaction logs for continuous improvement.
- **Embeddings Cache:** Maintains precomputed embeddings for faster similarity computation (using Redis or Pickle).

Fig. 4 shows Architechture Diagram.

VI. Model Implementation

Since Deliverable 2, several refinements were made to enhance usability, performance monitoring, system maintainability, and report clarity. While no major changes were made to architecture or preprocessing methods, improvements focused on interface layout, evaluation metrics, code organization, and documentation.

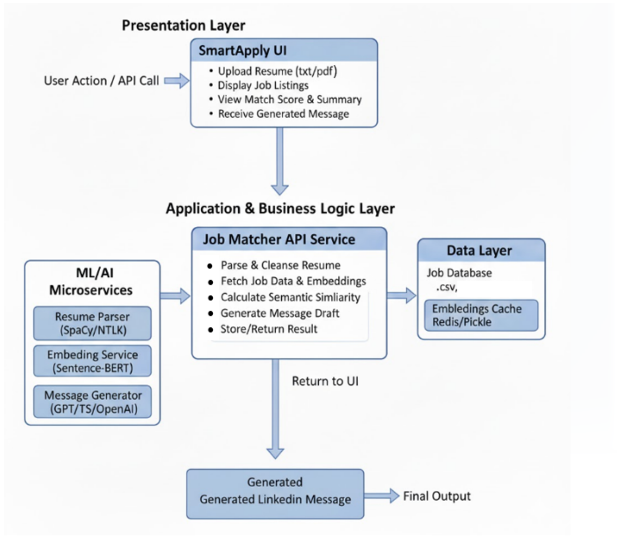

Presentation Layer
SmartApply UI
User Action / API Call
• Upload Resume (txt/pdf)
• Display Job Listings
• View Match Score & Summary
• Receive Generated Message
Application & Business Logic Layer
Job Matcher API Service
• Parse & Cleanse Resume
• Fetch Job Data & Embeddings
• Calculate Semantic Similarity
• Generate Message Draft
• Store/Return Result
ML/AI Microservices
Resume Parser (SpaCy/NTLK)
Embeding Service (Sentence-BERT)
Message Generator (GPT/T5/OpenAI)
Data Layer
Job Database .csv,
Embeddings Cache Redis/Pickle
Return to UI
Generated
Generated Linkedin Message
Final Output

Fig. 4. Architechture Diagram

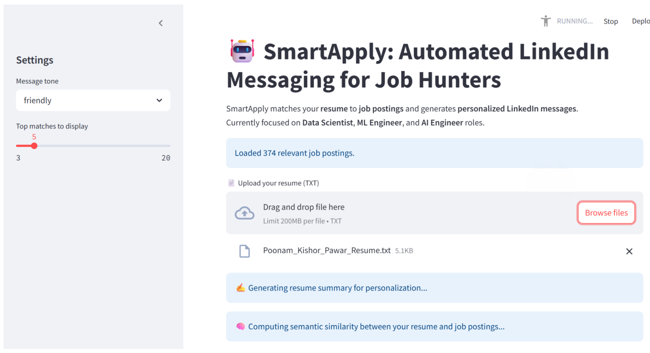
A. Models Explored

SmartApply leverages multiple NLP models to achieve end-to-end automation. Sentence embeddings are computed using SentenceTransformer ('all-MiniLM-L6-v2'), summarization is performed with Facebook's BART-large-CNN, and personalized messages are generated using Google's FLAN-T5-Large. The system runs on a local CPU environment with batch-encoded embeddings and similarity computation through cosine similarity from the SentenceTransformers library. Reproducibility is ensured by modularized code components in job_matcher.py and app.py.

B. Interface Layout and Functionality Enhancements

- The Streamlit UI now includes Top-N ranked job recommendations, similarity scores, and extracted matching skills.
- Users can now upload resumes, select tone, number of top matches, view ranked matches, and generate messages from a single interface.

Fig. 5 shows frontend UI of the app



Settings

Message tone: friendly

Top matches to display: 5 (3–20)

SmartApply: Automated LinkedIn Messaging for Job Hunters

SmartApply matches your **resume** to **job postings** and generates **personalized LinkedIn messages**.
Currently focused on **Data Scientist**, **ML Engineer**, and **AI Engineer** roles.

Loaded 374 relevant job postings.

Upload your resume (TXT)

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

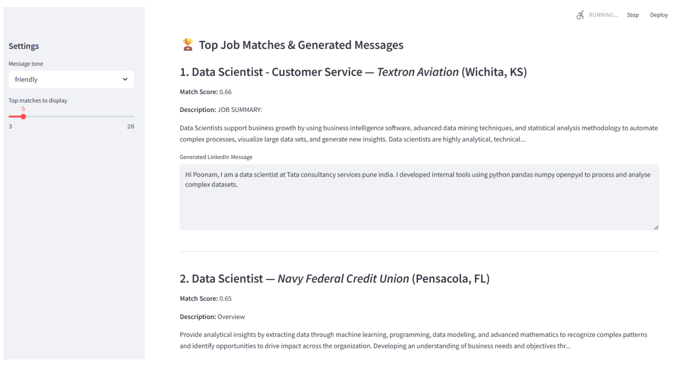
Poonam_Kishor_Pawar_Resume.txt 5.1KB

Generating resume summary for personalization...

Computing semantic similarity between your resume and job postings...

Fig. 5. Top 10 location for DS, ML, AI roles

Fig. 6 shows Autogenerated message


RUNNING... Stop Deploy

Settings

Message tone: friendly

Top matches to display: 5 (3–20)

Top Job Matches & Generated Messages

1. Data Scientist - Customer Service — *Textron Aviation* (Wichita, KS)

Match Score: 0.66

Description: JOB SUMMARY:

Data Scientists support business growth by using business intelligence software, advanced data mining techniques, and statistical analysis methodology to automate complex processes, visualize large data sets, and generate new insights. Data scientists are highly analytical, technical...

Generated LinkedIn Message

Hi Poonam, I am a data scientist at Tata consultancy services pune india. I developed internal tools using python pandas numpy openpyxl to process and analyse complex datasets.

2. Data Scientist — *Navy Federal Credit Union* (Pensacola, FL)

Match Score: 0.65

Description: Overview

Provide analytical insights by extracting data through machine learning, programming, data modeling, and advanced mathematics to recognize complex patterns and identify opportunities to drive impact across the organization. Developing an understanding of business needs and objectives thr...

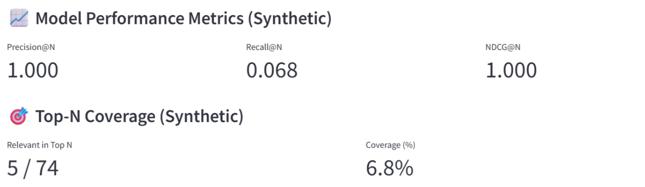
Fig. 6. Top 10 location for DS, ML, AI roles

C. Introduction of Evaluation Metrics

To strengthen the model assessment and enable performance comparison over time, the following ranking-based metrics were added:

- Precision@N, Recall@N, and NDCG@N
- Top-N Coverage
- Match Score Distribution
- Top Match Results

Fig. 7 shows the model performance metric


Model Performance Metrics (Synthetic)

Precision@N	Recall@N	NDCG@N
1.000	0.068	1.000

Top-N Coverage (Synthetic)

Relevant in Top N	Coverage (%)
5 / 74	6.8%

Fig. 7. Top 10 location for DS, ML, AI roles

Fig. 8 shows the performance graph

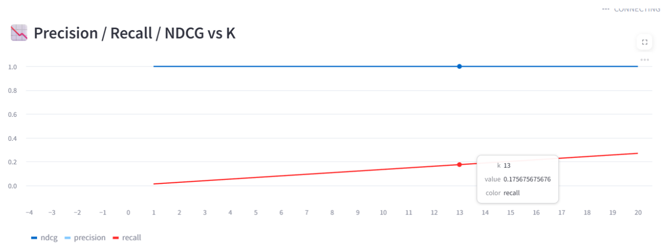


Fig. 8. Top 10 location for DS, ML, AI roles

Fig. 9 shows match score for top n jobs

Fig. 10 Top n jobs table

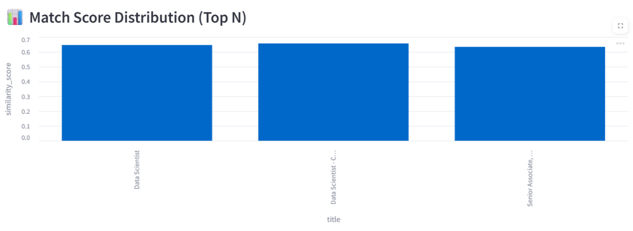
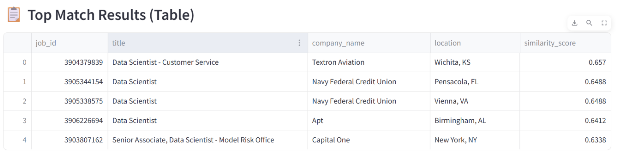


Fig. 9. Top 10 location for DS, ML, AI roles


Top Match Results (Table)

	job_id	title	company_name	location	similarity_score
0	3904379839	Data Scientist - Customer Service	Textron Aviation	Wichita, KS	0.657
1	3905344154	Data Scientist	Navy Federal Credit Union	Pensacola, FL	0.6488
2	3905338575	Data Scientist	Navy Federal Credit Union	Vienna, VA	0.6488
3	3906226694	Data Scientist	Apt	Birmingham, AL	0.6412
4	3903807162	Senior Associate, Data Scientist - Model Risk Office	Capital One	New York, NY	0.6338

Fig. 10. Top 10 location for DS, ML, AI roles

D. Repository and Codebase Organization

The project has been restructured into a modular GitHub repository for maintainability and reproducibility. Scripts (src), datasets (data), output files (results), UI components (ui), and documentation (docs) are now clearly separated, with version control implemented for tracking changes.

E. Documentation and Visual Improvements

- README updated with project summary, installation steps, execution guide, architecture overview, and sample outputs.
- Report now includes screenshots of the UI, Top-N ranking tables, and example generated messages.
- Added performance results comparisons and visual evidence for improved clarity.

F. Error Handling and Stability Updates

- Added try-except blocks in UI and backend for handling missing resume files, unsupported formats, and empty inputs.
- Added validation for file size, text extraction failures, and missing fields in job postings.
- Improved robustness of embedding generation by adding fallback mechanisms and logging.

VII. Challenges and Next Steps

A. Key challenges encountered include

- **Limited Semantic Understanding of Job Descriptions:**The current similarity-based ranking approach relies primarily on keyword matching and sentence embeddings, which struggle to fully capture contextual nuances like role responsibilities, domain expertise, and experience depth.
- **Lack of Ground Truth Labels for Evaluation:**The system evaluates match quality using synthetic test cases

since real recruiter feedback or labeled job-resume relevance data is unavailable, limiting the reliability of performance metrics.

- **No Real-Time Integration with LinkedIn or Job Portals:**Due to API restrictions, live scraping and integration with real job postings or LinkedIn messaging are not yet supported. The system currently relies on static offline datasets.

B. Next Steps and Future Enhancements

- **Enhance Message Generation with GPT-based Prompt Engineering:** Move from static templates to guided prompt-based content generation for better personalization, tone control, and context awareness.
- **Improve System Scalability and Deployment with Docker/Streamlit Cloud:** Containerize the system and deploy for multi-user access, enabling recruiters and job seekers to use SmartApply in real-world settings.
- **Long-Term Vision:**The ultimate goal is to transform SmartApply into a fully deployable AI-based career recommendation assistant that supports real-time job matching, integrates with professional platforms, generates ethical and personalized messages, and continuously improves based on user feedback.

VIII. Responsible AI Reflection

SmartApply emphasizes responsible AI practices by avoiding bias amplification in text generation. All user resumes are processed locally without storage to preserve privacy. Generated recruiter messages are filtered to maintain professionalism and non-discriminatory language. While SmartApply introduces automation efficiencies in job matching and outreach messaging, responsible AI practices are essential to ensure fairness, protect user data, and maintain trust. The system currently applies mitigation strategies like anonymization, controlled message templates, and transparency indicators, with future plans focused on explainability, data encryption, and fairness auditing.