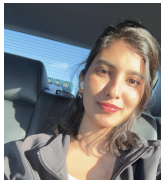
ResumeGenie: Intelligent Resume Optimization with LLMs

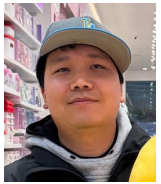
Aryan Vats

MS in CS



Nidhi Choudhary

MS in ADS



Eben Gunadi
MS in DS
(Healthcare)

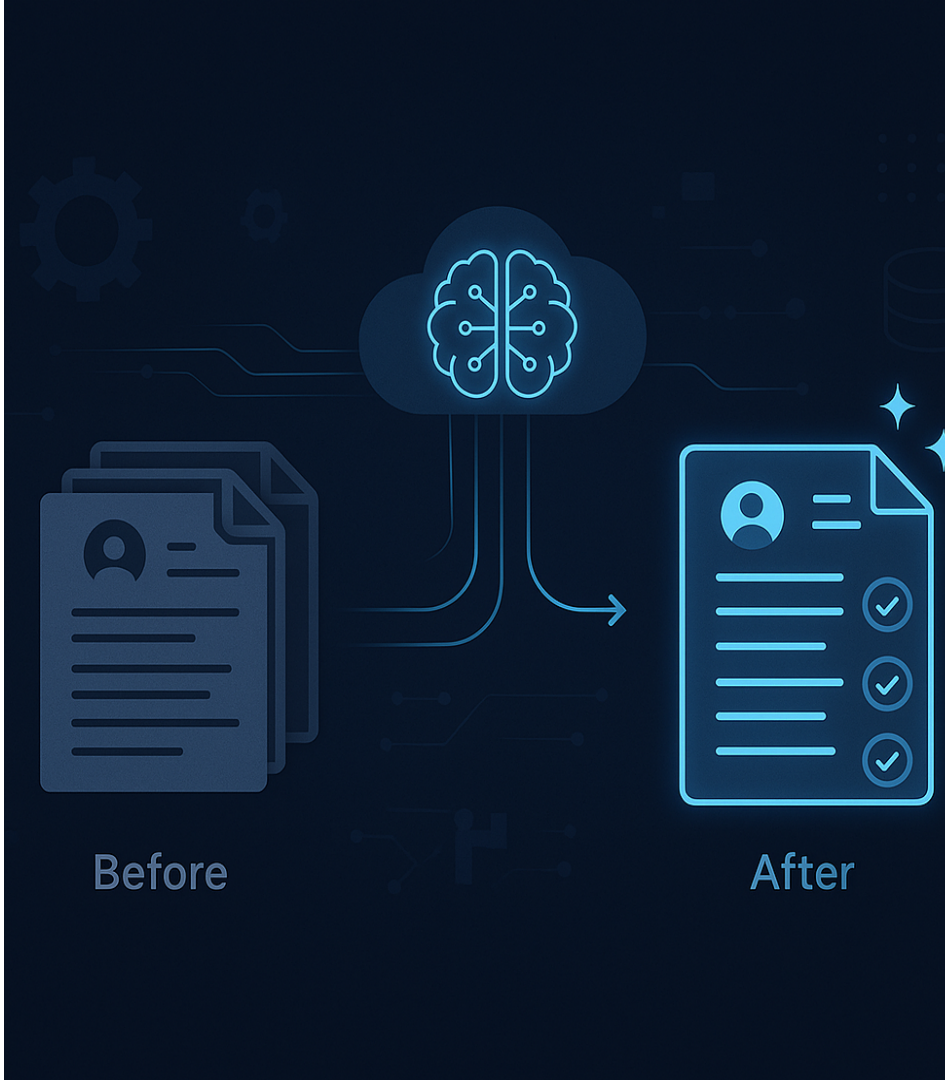
Muqi Zhang

MS in CS

Justin Chen

BS in DS

Table of Contents



Why are we here?

Situation

The **job market** is highly **competitive**, and landing the right role is tougher than ever. On top of that, Applicant Tracking Systems (**ATS**) **filter out many resumes** before they even reach a recruiter.

Candidates need strong, ATS-compliant resumes to even get noticed.

Complication

Most resumes fail to stand out due to generic content and lack of strategic positioning.
Creating a **structured**, **impactful**, and **keyword-optimized** resume for every application is **critical** — but time-consuming.
Manually tailoring resumes to each job description is unrealistic when candidates need to apply to dozens of roles quickly.

Key Question

How can we leverage technology, especially LLMs, to **automate and personalize resume generation** – making every resume tailored, ATS-friendly, and impactful – to **maximize a candidate's chances** across multiple job applications?

Solution Orchestration

Solution Orchestration

Candidate Upload: Raw input files (resume PDFs and Job Descriptions)

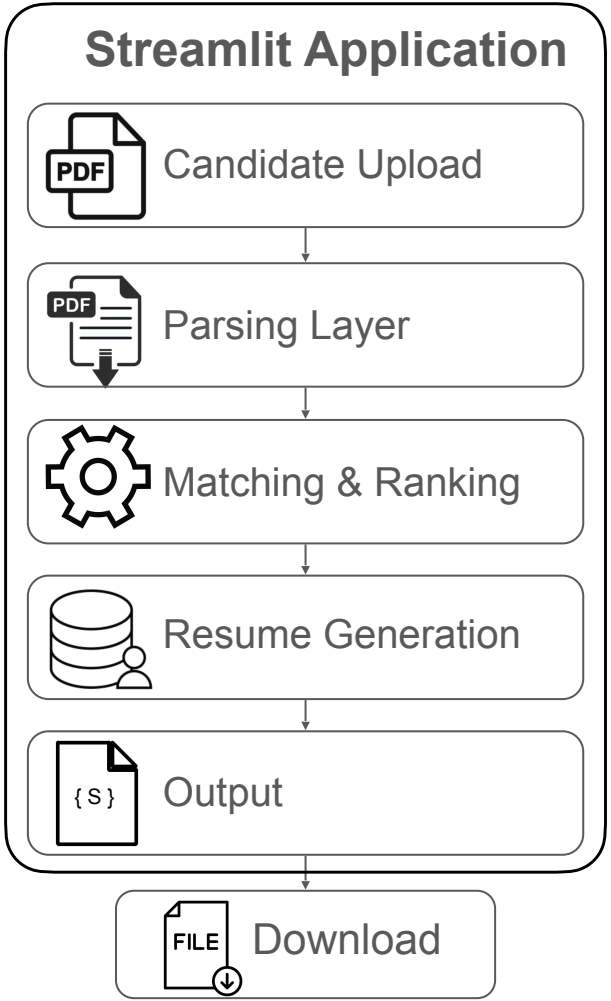
Parsing Layer: Resume parsing (PDF → JSON)

Prompt Engineering: Craft structured prompts for LLM based on parsed data

Matching & Ranking: Search and match using Vector Database (resume embeddings vs. job description embeddings)

Resume Generation: Fine-tuned resume generated (using LLM with engineered prompts)

Output: Downloadable final resume (JSON → PDF)



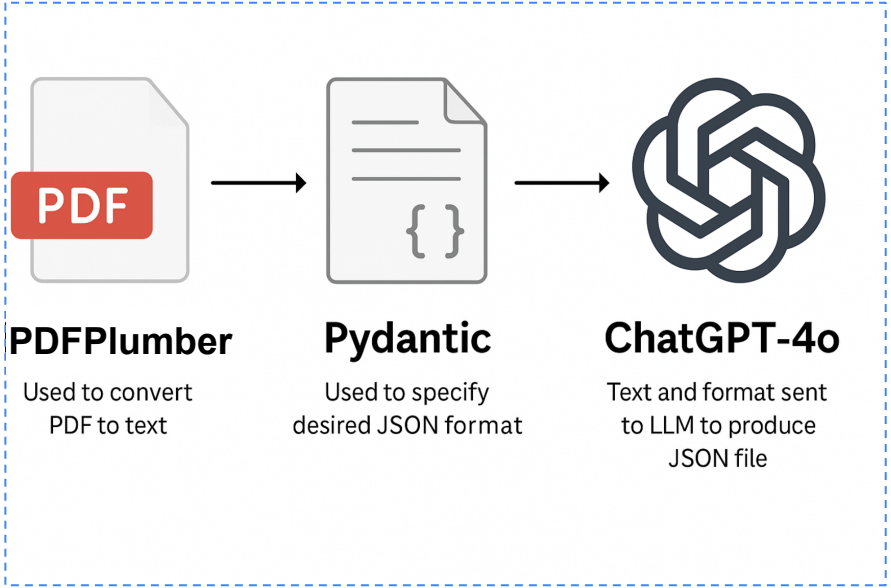
Deep Dive into the Solution Workflow

Parsing with PDFPlumber, Pydantic and ChatGPT-4o

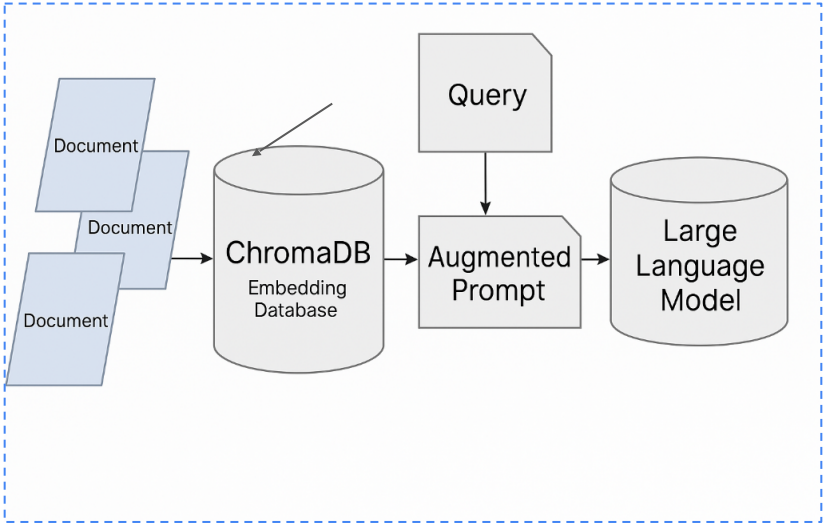
- **PDFPlumber**
- Used to convert PDF to text

- **Pydantic**
- Used to specify desired JSON format
- Text lines get organized neatly into boxes { key: value }

- **ChatGPT-4o**
- Finally, text and format sent to LLM to produce JSON file
- Neatly formatted JSON file produced by the AI



ChromaDB for Retrieval Augmented Generation (RAG)



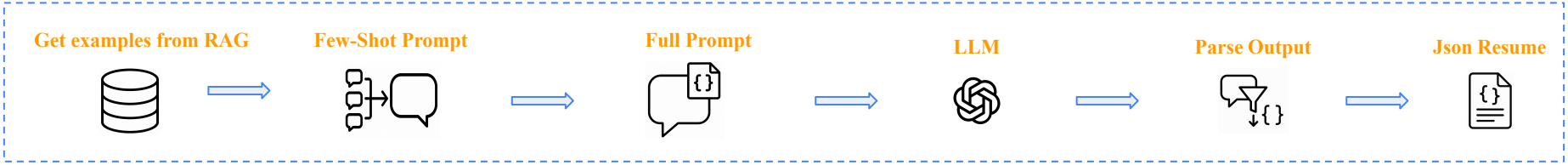
RAG Architecture

- Create a knowledge base for the LLM to reference
- Reduce hallucinations by grounding responses to the retrieved information

ChromaDB

- Open source vector database
- Stores data as numerical vectors with embeddings
- Quickly queries relevant documents with similarity metrics

Langchain Framework Flow



Input Chain: Raw resume is ingested through a Chain with a custom **Prompt Template** to extract the major/field.

Retriever Query: A Retriever searches ChromaDB for **top 2 similar resumes** based on the extracted field.

Few-Shot Prompt Assembly: Retrieved example resumes are inserted into a **Few-Shot Prompt Template** to condition the LLM.

Resume Generation Chain: The LLM generates an **ATS-friendly resume** using the few-shot examples and original input.

Output Parsing: An **Output Parser** ensures the generated resume has structured fields.

Application Walkthrough and Result

Comparing Results

John Doe Los Angeles, CA
johndoe@usc.edu +1 (213) 999-9999

EDUCATION

University of Southern California, Master of Science - Computer Science May 2025
Analysis of Algorithms, Deep Learning, Machine Learning for Data Science
University of Pittsburgh, Bachelor of Science - Mathematics and Economics July 2022
Probability & Statistics, Linear Algebra, Applied Econometrics, Game Theory, Calculus 1-3

WORK EXPERIENCE

Mclearn Racing Indianapolis
Data Engineer August 2022 - July 2023

- Handled the receipt, cleaning, and preparation of data from clients using SAS, SQL, and Excel to assist data scientists in building marketing mix models that contributed to improved performance.
- Created a program in SAS that automated the refinement of linear regression models for different customer segments, streamlining routine processes and enhancing team efficiency.

SKILLS

- Programming: SAS (base SAS and Macros), SQL
- Supervised Learning: linear and logistic regressions, decision trees, support vector machines (SVM)
- Unsupervised Learning: k-means clustering, principal component analysis (PCA)
- Data Visualization: Excel, Google Sheets

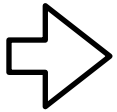
PROJECTS

Fantasy F1 Modeling

- Wanted to stop losing at Fantasy F1 Manager so I aggregated and prepped 5 years of fantasy projection data into a MySQL database
- Built a random forest model in SAS that synthesized the various projections into a unified forecast, leading to improved predictive accuracy compared to other available methods.
- I had heck ton of points on it relative to my previous methodology

Strategy Recommendation Engine

- Aggregated data from F1nsight API and used k-nearest-neighbors in SAS to build a better recommendation system for making me predict wins
- I made better pitstop strategies while I play F1 Racing compared my previous methodology, making me win races.



John Doe

johndoe@usc.edu | | Los Angeles, CA

SUMMARY

Data scientist with expertise in SAS, SQL, and machine learning algorithms. Experienced in developing predictive models and data visualization techniques.

WORK EXPERIENCE

Data Engineer at Mclearn Racing
2022-08 – 2023-07

Handled the receipt, cleaning, and preparation of data from clients using SAS, SQL, and Excel to assist data scientists in building marketing mix models that contributed to improved performance.

Developed and implemented a program in SAS to automate the refinement of linear regression models for diverse customer segments, reducing processing time by 30% and enhancing team productivity. Collaborated with stakeholders to understand data requirements and built pipelines for data ingestion into the data lake.

EDUCATION

University of Southern California - Master of Science in Computer Science
– 2025-05
Analysis of Algorithms, Deep Learning, Machine Learning for Data Science

University of Pittsburgh - Bachelor of Science in Mathematics and Economics
– 2022-07
Probability & Statistics, Linear Algebra, Applied Econometrics, Game Theory, Calculus 1-3

SKILLS

Programming
SAS (base SAS and Macros) SQL Python C#

Machine Learning
Linear and logistic regressions Decision trees Support vector machines (SVM) Predictive analytics

Data Visualization
Power BI Excel Google Sheets

PROJECTS

Fantasy F1 Modeling

Aggregated and prepped 5 years of fantasy projection data into a MySQL database and built a random forest model in SAS.

Enhanced predictive accuracy by 20% compared to existing methods, resulting in a 15% increase in points relative to previous methodology.

Strategy Recommendation Engine

Aggregated data from F1nsight API and used k-nearest-neighbors in SAS to build a better recommendation system for predicting wins.

Optimized pitstop strategies in F1 Racing, contributing to a 10% increase in race wins.

GRIDS

THANK YOU

ress-gen.streamlit.app

This is my script for page5 (solution orchestration):
The candidate first uploads their raw resume and the corresponding job description through our web interface.
On the backend, we immediately capture these files and start parsing. We use PDFPlumber to extract raw text from the PDFs, Pydantic to organize the extracted data into a structured JSON schema, and then leverage ChatGPT-4o to further refine and format the JSON output cleanly.
After parsing, we move into the Retrieval-Augmented Generation (RAG) step. Using ChromaDB, we store and query resume embeddings to retrieve the top two most relevant resume examples based on the candidate's field.
These retrieved examples are then fed into a LangChain pipeline. We construct a few-shot prompt by combining the retrieved examples with the original parsed data to guide the LLM more accurately during resume generation.
The LLM then generates a fine-tuned, ATS-optimized resume for the candidate.
Finally, the generated resume is displayed on the web application and can be downloaded by the candidate in PDF format.

my script for pages 7-8:
parsing with pdfplumber, pydantic, and chatgpt-4o)
after the user uploads their PDF, we first extract the PDF text using PDFPlumber. to convert this text into JSON format, we use pydantic to specify the desired format or key-values. then both the text and this format is sent to ChatGPT to produce the final JSON file.
chromadb for retrieval augmented generation
the JSON extracted from the user's PDF will then updated with a strategy called retrieval augmented generation, also known as RAG. we create a knowledge base of tried and true resumes that the LLM references when augmenting our JSON. this grounds the final output to the retrieved information and reduces hallucinations. for reference, the knowledge base we used is a vector database called chromaDB. vector databases store data as numerical vectors with embeddings, and can query relevant documents by finding the similarity between those vectors.

page 9:
On this slide, you can see the flow we've built using LangChain to transform a raw resume into an ATS-ready JSON document. Let's walk through each step.
Input Chain
First, we ingest the candidate's raw resume into an Input Chain. This Chain applies a custom PromptTemplate to extract and identify applicant's major or field of expertise—which we'll use to drive our retrieval logic.
Retriever Query
Next, that extracted field powers a Retriever connected to a ChromaDB, where the vector stores. We perform a similarity search to fetch the top two examples of domain-matched resumes from our RAG database. These examples will serve as our few-shot conditioning example to demonstrate to the LLM what is a success change to make it ATS-ready.
Few-Shot Prompt Assembly
With those retrieved resumes in hand, we assemble a FewShotChatMessagePromptTemplate. This template allows the LLM to understand what a good change looks like.
Resume Generation Chain
We then pass the combined prompt—system instruction, few-shot examples, and the candidate's raw resume—into a second Chain, powered by ChatOpenAI. The LLM generates an ATS-compliant JSON resume, using the patterns illustrated in our examples.
Output Parsing
Finally, an Output Parser cleans and validates the LLM's response, ensuring the result is well-formed JSON with structured fields.