**Team Members:**

Janani Karipireddy - 2282381

Tanmai Bhashyam - 2307181

Varshini Kotha - 2307321

AI Powered Resume Reviewer & Enhancer: Cohere Model

**CSCI 4391/5931: Term Paper**



|  |  |
| --- | --- |
| + |  |

**Course Instructor:** Dr.Khondker Shajadul Hasan

Table of Contents

[INTRODUCTION 2](#_Toc196393592)

[MODEL’S ARCHITECTURE 2](#_Toc196393593)

[HOW DOES THE MODEL WORKS? 3](#_Toc196393595)

[TRAINING PROCESS FOR THE MODEL 4](#_Toc196393597)

FINE-TUNING PROCESS FOR THE MODEL [5](#_Toc196393598)

PARAMETER EFFICIENT FINE-TUNING PROCESS FOR THE MODEL [6](#_Toc196393598)

CODE DEMONSTRATION [7](#_Toc196393598)

OUTPUTS [11](#_Toc196393598)

BENEFITS AND USEFULNESS OF THE MODEL [13](#_Toc196393598)

[CONCLUSION 14](#_Toc196393599)

WORKLOAD DISTRIBUTION AMONG GROUP MEMBERS [14](#_Toc196393598)

[REFERENCES 14](#_Toc196393608)

# **INTRODUCTION**

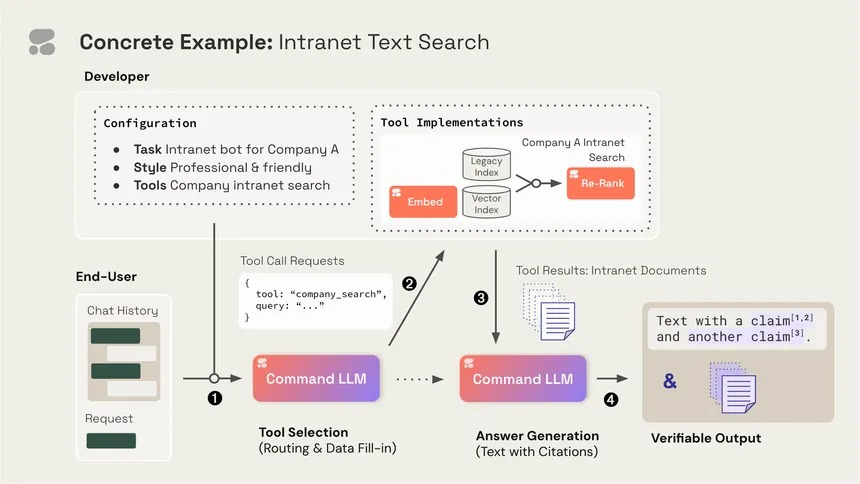
Large Language Model’s (LLM’s) introduction has had a big impact on career development, particularly on resume writing. In the past, job seekers were forced to either use trial-and-error methods or hire professional resume writers to enhance their resumes. This procedure is getting more intelligent and accessible with the advent of AI solutions like the Resume Chatbot. The main goal of our project is to assess, improve, and offer feedback on resumes using Coherer’s LLM. Assisting users in properly showcasing their qualifications is the aim. Both text and PDF resumes are accepted by the system, which can also take certain job descriptions into account which would be included in PDF resumes or text. Context-aware recommendations are made possible as a result, increasing the relevance of resumes to job applications.

Through the interactive chatbot interface, customers can ask for improvement ideas or career assistance. The goal of this innovation is to close the gap between regular users and professional resume writing services. Additionally, it demonstrates how AI technologies are being used in the real world to improve employability. We offer an easy-to-use and effective application that gives people data-driven career guidance.

# **MODEL’S ARCHITECTURE**

The Resume Chatbot's core design is based on a client-server architecture, with Cohere's API for backend model inference and Streamlit for the user interface. The frontend is a small web application built using Python that manages task selection, file uploads, and output display. Resumes can be entered by users using a text field or a PDF upload, and a job description is optional. After processing, these inputs are transformed into prompts. Cohere's "command-r-plus" model receives the prompt on the backend and uses transformer-based natural language processing to produce a response. The model itself is built on a decoder-only architecture that is optimized for text creation and instruction following. To extract contextual meanings and produce pertinent reactions, it internally makes use of several layers of attention mechanisms.

Additionally, the chatbot maintains a session state to record previous conversations. Users can ask the AI questions about job roles, resume assistance, or general career planning thanks to this, which makes it possible for cohesive, multiturn interactions. Because each component is modular, it can be scaled in the future to accommodate various frontend designs or LLM’s. Low latency and good interpretability are given top priority in the design to provide a responsive and clear user experience. Finally, the architecture has download options, allowing users to store AI responses for later editing or offline use.

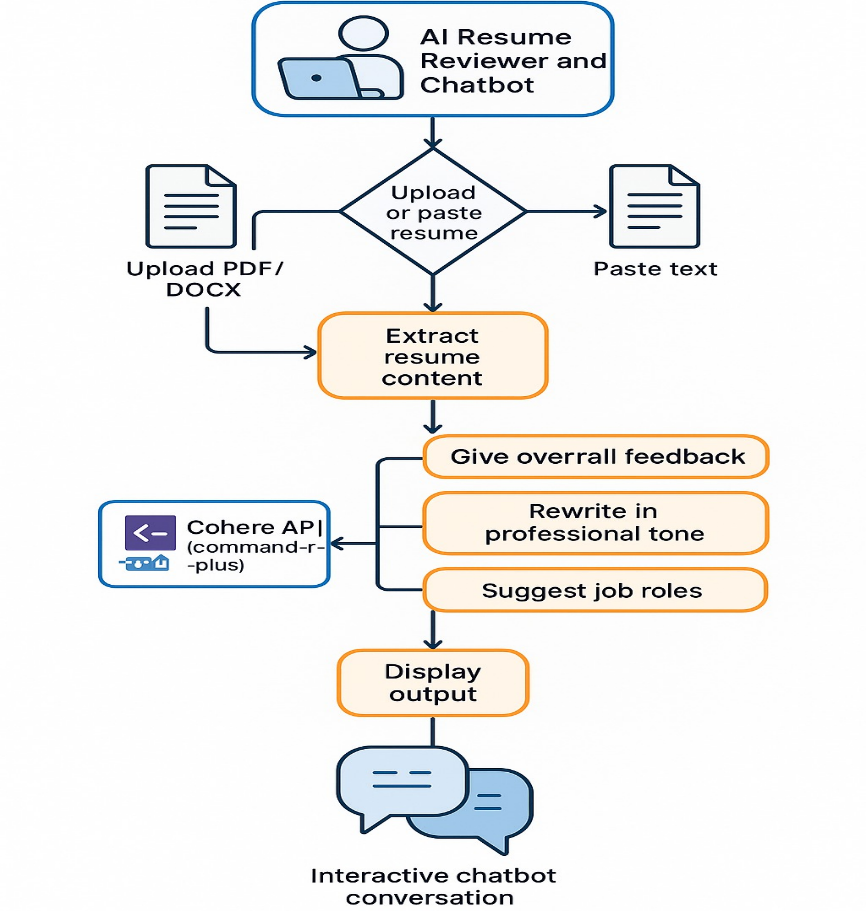


# **Figure 1** - Command R+ Model Architecture

# **HOW DOES THE MODEL WORKS?**

This Streamlit app acts as an AI-powered resume enhancer and career advisor by using Cohere’s command-r-plus language model, a powerful large language model trained to understand and generate human-like text. When a user uploads a resume (PDF) or pastes the text, the app extracts the content and, depending on the selected tasks such as "Give overall feedback," "Rewrite in a more professional tone," or "Suggest suitable job roles” the app builds a dynamic prompt. This prompt is a carefully crafted instruction that includes both the task and the full resume text (and optionally a job description). The prompt is then sent to Cohere’s model via the co.generate() function. The model processes this prompt, understands the user's intent, analyzes the resume content, and generates a detailed, context-specific response. The model uses up to 10,000 tokens, which allows it to handle long documents like full resumes with ease.

For the chatbot functionality, the model is used to simulate a conversation with a virtual career advisor. Every time the user sends a question (e.g., "How can I improve my resume for data science roles?"), the app builds a conversation history prompt that includes the full resume, past messages, and the new user input. This prompt is again sent to Cohere, which interprets the entire conversation in context and returns a helpful, relevant reply. The app displays these exchanges in a chat-like format and stores the history so that each new question builds upon the last. This conversational design enables the model to offer dynamic, context-aware responses, making it feel like you're chatting with a knowledgeable career coach who has thoroughly read your resume. In essence, the model powers both one-time enhancements and continuous interactive guidance—all driven by smart prompt engineering and Cohere’s natural language capabilities.

****

# **Figure 2** - Resume Chatbot Workflow Employing Cohere Model Integrating with Streamlit

# **TRAINING PROCESS FOR THE MODEL**

The training of the command-r-plus language model is a multi-phase process that begins with pretraining on massive and diverse text datasets. These datasets include billions of words collected from books, websites, articles, technical manuals, resumes, job descriptions, and more. The goal of this stage is to teach the model the fundamentals of language—grammar, sentence structure, vocabulary, and meaning—by exposing it to countless examples of real-world writing. During pretraining, the model learns using a technique called unsupervised learning, where it tries to predict the next word in a sentence based on the previous ones. For instance, if it sees the phrase "experienced in project," it learns that "management" might be a common next word. Through millions of these predictions, the model develops a nuanced understanding of how language works in different domains and contexts.

After pretraining, the model is further fine-tuned using supervised learning and reinforcement learning from human feedback (RLHF). In this stage, it’s exposed to specific tasks—such as answering questions, rewriting text, or offering step-by-step guidance—along with ideal responses provided by human trainers. This allows the model to learn how to follow instructions, stay focused on the task, and provide useful and accurate outputs. Reinforcement learning helps improve the quality of its responses over time by rewarding the model for outputs that are rated highly by humans. This fine-tuning process essentially turns the model from a general-purpose language engine into a responsive assistant capable of handling detailed, domain-specific instructions—like rewriting a resume or giving career advice.

For tasks related to resumes and careers, the model benefits significantly from exposure to professional and employment-related documents. During training, it sees many examples of resumes, CVs, cover letters, LinkedIn summaries, and job postings across different industries. This helps the model learn what makes a resume strong, how to express accomplishments effectively, and how certain skills align with different job titles or industries. When a user submits their resume and chooses a task—like improving tone, tailoring it to a job description, or finding suitable job roles—the model uses its learned knowledge to generate a relevant and context-aware output. It doesn’t memorize specific resumes but rather generalizes patterns it has seen across its training data to produce personalized, high-quality responses based on the unique input it receives

**FINE-TUNING PROCESS FOR THE MODEL**

After the initial pretraining stage—where the model learns the structure and general use of language from a massive dataset—the fine-tuning process narrows the model’s focus to specific, real-world tasks. Fine-tuning is done using smaller, curated datasets that contain labeled examples. These examples often include inputs (like prompts or instructions) paired with ideal outputs written by human experts. This stage helps the model learn how to perform tasks such as rewriting text professionally, answering specific questions, or providing structured feedback. Essentially, fine-tuning adapts a general-purpose language model into a more task-specialized assistant, capable of understanding and responding to complex instructions with precision.

In many cases, especially for advanced models like Cohere’s command-r-plus, the fine-tuning process also incorporates a technique called Reinforcement Learning from Human Feedback (RLHF). This involves showing the model multiple possible answers to a prompt, having human trainers rank those answers, and then training the model to prefer the highest-rated responses. A reward model is built to learn these preferences, and the main model is trained to generate outputs that align with this reward signal. This step significantly improves the model’s ability to respond in ways that are helpful, accurate, aligned with user expectations, and aligned with human values—especially important in sensitive contexts like resume advice or career guidance.

Fine-tuning is what enables the model to go beyond just understanding language to actually performing useful, specialized tasks. For example, when a user submits a resume and asks for feedback, the model doesn’t just summarize or rephrase—it evaluates based on patterns and best practices it has learned from fine-tuning on professional content. It can suggest improvements in tone, recommend better phrasing, or tailor the content to a job description, all because it was trained to recognize and respond to these types of instructions. Without fine-tuning, the model would be much more generic and less capable of adapting its responses to the specific needs of job seekers or professionals.

**PARAMETER EFFICIENT FINE-TUNING PROCESS**

# For the Resume chatbot application using Cohere’s command-r-plus model, Parameter-Efficient Fine-Tuning (PEFT) can be applied to optimize the model’s ability to handle specific tasks, such as analyzing resumes, giving feedback, or suggesting job roles. Here's what could fall under the PEFT process for your application:

**1. Adapter Layers for Task-Specific Modifications**

In your application, adapter layers could be used to tailor the model’s behavior for specific resume-related tasks. For example, one adapter could focus on improving the model’s ability to give feedback on resumes, another could specialize in suggesting suitable job roles, and a third could be dedicated to rewriting resumes in a professional tone. Each adapter would be trained to handle a specific task while keeping the rest of the model frozen, which means the general language capabilities remain intact. This allows your chatbot to provide more targeted and effective responses to users, while avoiding unnecessary resource consumption by updating the whole model.

**2. LoRA (Low-Rank Adaptation) for Efficient Learning**

Using LoRA (Low-Rank Adaptation) in your resume chatbot application could significantly reduce the number of parameters that need to be updated. Instead of fine-tuning the entire model’s weights, LoRA adds small, efficient low-rank matrices into the model’s attention mechanism. This would help the chatbot better adapt to nuanced resume-specific tasks like improving phrasing, identifying strengths or weaknesses in a resume, or matching a resume with job descriptions—all while keeping the base model’s core knowledge intact. LoRA makes this process faster and more efficient, enabling the chatbot to learn task-specific patterns without the computational overhead of training a whole model.

**3. Prompt Tuning for Task Customization**

Prompt tuning can be especially useful in your resume chatbot application. In this case, the model could learn to adjust its behavior based on specific learned input prompts (soft prompts) that influence the way it processes and responds to a resume. For example, a task-specific prompt could guide the model to focus on the professional tone of the resume, while another prompt could instruct it to identify potential job roles based on skills and experience listed in the resume. These prompts would be trained separately from the model, allowing for quick adaptations to the chatbot’s behavior for different user needs without altering the model’s core functionality.

**USING THE MODEL FOR RESUME CHATBOT APPLICATION**

The integrated chatbot also allows users to ask follow-up questions, like "What kind of roles fit my experience in project management?" Or "How should I improve my summary section?" While using the same API, each interaction is directed by unique prompt engineering. Because of its adaptability, the tool can be used for career planning as well as resume writing. Furthermore, it feels more like a dialogue with a career counselor when you may reply to contextual questions. These illustrations show the model's versatility and intelligence across a range of resume-related processes.

**CODE DEMONSTRATION**

!pip install cohere streamlit pdfplumber

!pip install python-dotenv

# %%

%%writefile resume\_app\_chatbot.py

import streamlit as st

import cohere

import pdfplumber

# Initialize Cohere client

co = cohere.Client("q3dy0ah5ODJBkbHZoNvT5WbSg8TUsbpiFllJaIOs")

st.set\_page\_config(page\_title="AI Resume Reviewer", layout="wide")

st.title("AI-Powered Resume Reviewer & Enhancer")

# Sidebar: Resume Upload or Text Input

st.sidebar.header("Upload or Paste Resume")

uploaded\_file = st.sidebar.file\_uploader("Upload a resume (PDF or DOCX)", type=["pdf", "docx"])

resume\_text\_area = st.sidebar.text\_area("Or paste your resume text:", height=250)

st.sidebar.markdown("---")

st.sidebar.header("Optional")

job\_description = st.sidebar.text\_area("Paste job description (optional)", height=150)

# Extract resume text

resume\_text = ""

if uploaded\_file:

    with pdfplumber.open(uploaded\_file) as pdf:

        resume\_text = "\n".join([page.extract\_text() for page in pdf.pages if page.extract\_text()])

elif resume\_text\_area.strip():

    resume\_text = resume\_text\_area.strip()

# Enhancement Options

st.subheader("Choose Enhancement Type")

task = st.selectbox(

    "Select what you'd like the assistant to do:",

    [

        "Give overall feedback",

        "Rewrite in a more professional tone",

        "Suitable job roles"

    ]

)

if st.button("Enhance Resume"):

    if not resume\_text:

        st.warning("Please upload a resume or paste the text.")

    else:

        prompt = ""

        if task == "Give overall feedback":

            prompt = f"Provide detailed and constructive feedback on the following resume:\n\n{resume\_text}"

        elif task == "Rewrite in a more professional tone":

            prompt = f"Rewrite the following resume to sound more polished and professional:\n\n{resume\_text}"

        elif task == "Suitable job roles":

            prompt = f"Analyze the following resume and suggest a list of suitable job roles for the candidate:\n\n{resume\_text}"

        if job\_description and task != "Suitable job roles":

            prompt += f"\n\nThe resume should be tailored to this job description:\n{job\_description}"

        with st.spinner("Analyzing and enhancing your resume..."):

            response = co.generate(

                model="command-r-plus",

                prompt=prompt,

                max\_tokens=10000,

                temperature=0.7

            )

            output\_text = response.generations[0].text.strip()

        st.subheader("AI Output")

        st.write(output\_text)

        st.download\_button(

            label="Download Output",

            data=output\_text.encode("utf-8"),

            file\_name="resume\_output.txt",

            mime="text/plain"

        )

# 🤖 Resume Chatbot Section

st.markdown("---")

st.header("💬 Chat with Resume AI Assistant")

# Initialize chat history

if "chat\_history" not in st.session\_state:

    st.session\_state.chat\_history = []

user\_input = st.text\_input("Ask a question about your resume, career path, or job suggestions:")

if user\_input and resume\_text:

    # Build conversation context

    chat\_prompt = "You are an expert resume and career advisor AI. Use the following resume to answer the user's questions.\n"

    chat\_prompt += f"\nResume:\n{resume\_text}\n"

    for pair in st.session\_state.chat\_history:

        chat\_prompt += f"\nUser: {pair['user']}\nAI: {pair['ai']}"

    chat\_prompt += f"\nUser: {user\_input}\nAI:"

    # Call Cohere

    with st.spinner("Thinking..."):

        chat\_response = co.generate(

            model="command-r-plus",

            prompt=chat\_prompt,

            max\_tokens=10000,

            temperature=0.7

        )

        ai\_reply = chat\_response.generations[0].text.strip()

    # Update history

    st.session\_state.chat\_history.append({"user": user\_input, "ai": ai\_reply})

# Display chat messages

if st.session\_state.chat\_history:

    for chat in st.session\_state.chat\_history:

        st.markdown(f"\*\*You:\*\* {chat['user']}")

        st.markdown(f"\*\*AI:\*\* {chat['ai']}")

# %%

!streamlit run resume\_app\_chatbot.py

# %%

**OUTPUTS**

**A screenshot of a computer

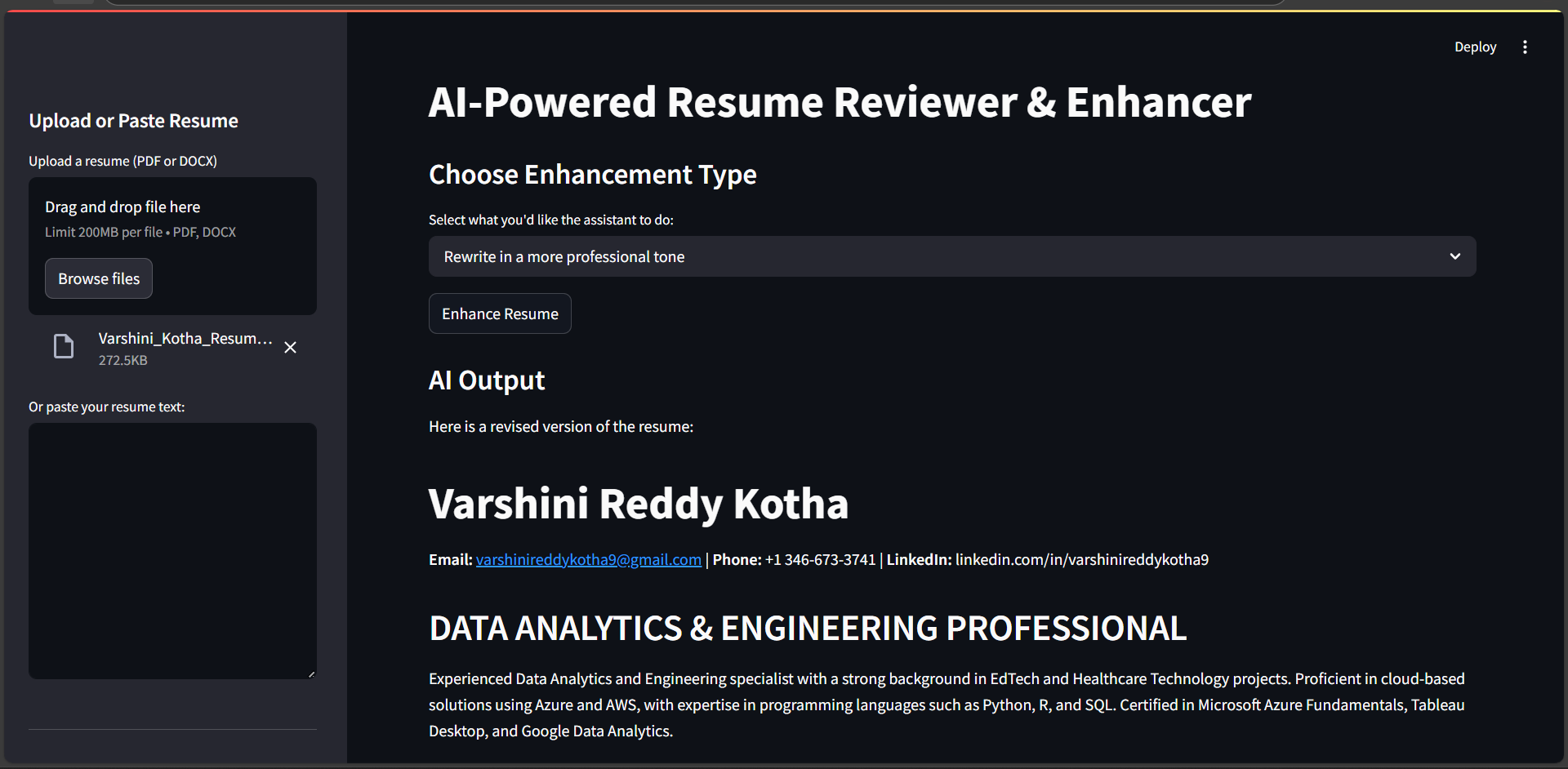
AI-generated content may be incorrect.**

**Figure 3.** AI-Powered Resume Reviewer & Enhancer Interface and Listing of Tasks

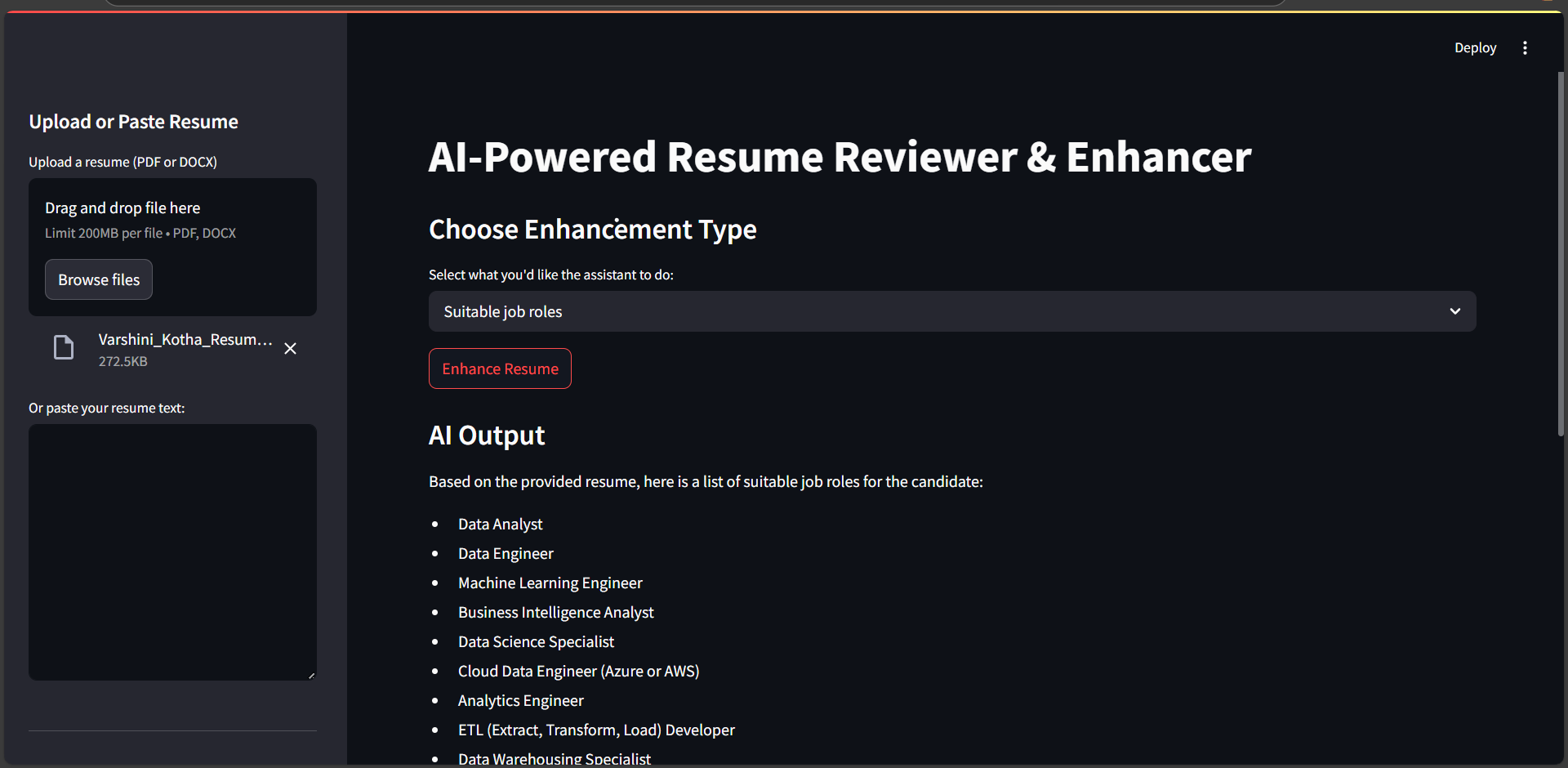
**A screenshot of a computer program

AI-generated content may be incorrect.**

**Figure 4**. Selection of Task “Give Overall Feedback” and Providing Feedback on the uploaded Doc/PDF



**Figure 5.** Selection of Task “Rewrite in a more Professional tone” and Generation of Updated Resume in ATS Format



**Figure 6.** Selection of Task “Suitable Job Roles” and Suggesting Roles based on uploaded Resume

A screenshot of a chat

AI-generated content may be incorrect.

**Figure 7.** Interactive Chat-bot Conversation By providing Prompt related to the above tasks

**BENEFITS AND USEFULNESS OF THE MODEL**

The Resume Chatbot offers job seekers and career changers several benefits. First, it provides real-time, AI-powered feedback that can greatly improve a resume's quality. As a result, expensive professional writing services are no longer required. Second, the chatbot facilitates individualized communication, enabling consumers to ask questions or look for further recommendations. Third, by including job descriptions, it facilitates context-aware rewriting and helps customize resumes for certain positions. Additionally, the approach is simple to use—no technical knowledge or coding is needed. We make sure the program works on every device with a browser by utilizing Streamlit.

The model's responses are also professional and well-structured, making them appropriate for use directly in job applications. The AI output can also be downloaded, which facilitates examination, editing, and reuse. By automating formatting, wording, and structure activities, the application also saves time. Lastly, its instructional value is noteworthy: by reading AI recommendations, people may learn how to make their resumes better. The chatbot's twin function of automation and learning support makes it exceptionally advantageous.

# **CONCLUSION**

This research shows how sophisticated AI models can be used to improve and streamline the resume-writing process. Cohere's LLM is combined with an easy-to-use Streamlit interface to provide a potent career development tool. Users may upload their resumes, get personalized feedback, and have AI-powered discussions to get their questions answered. Because of its adaptability, the model can be used for a variety of purposes, such as rewriting text or identifying job roles. We make sure the system produces pertinent and useful results by using pretrained and instruction-tuned models. Additionally, future model modifications, simple updates, and interaction with external APIs are all supported by modular architecture.

This initiative demonstrates the practicality of AI applications outside of researching situations that have an immediate influence on user lives. Additionally, it creates opportunities for additional innovation, including incorporating resume scoring, supporting several languages, or optimizing applicant tracking systems (ATS). To sum up, our Resume Chatbot integrates usability, intelligence, and accessibility into a single platform with the goal of enhancing professional documents and assisting users during their job search.

**WORKLOAD DISTRIBUTION AMONG GROUP MEMBERS**

|  |  |
| --- | --- |
| Members | Workload Distribution |
| Member 1 | Frontend & UI Developer |
| Member 2 | Backend & ML Model Developer |
| Member 3 | Documentation, Testing |

# **REFERENCES**

1. <https://www.acorn.io/resources/learning-center/cohere-ai/>
2. Cohere Documentation: <https://docs.cohere.com/>
3. Streamlit Documentation: <https://docs.streamlit.io/>
4. Brown, T., et al. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165.
5. Raffel, C., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research.
6. Vaswani, A., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.