Blaize Benson
EGR 404 – Generative AI Tools
5 May 2025
Sendag Resit, Watkins Justin
GitHub Repository: https://github.com/blaizebenson/ResumeGPT2
Live App: https://resumegpt2-4rikzklkqx3btcpxumsbib.streamlit.app/

ResumeGPT2 – Final Report

1. Introduction

Cover letters remain a standard part of the job application process. Although resume optimization and automated systems like ATS (Applicant Tracking Systems) are widely used, most hiring managers still expect a personalized cover letter that articulates why the applicant is a good fit for the position. However, writing unique, high-quality letters for each job can be a tedious and inefficient process, especially for applicants applying to many positions. The emergence of generative AI tools like GPT-4 presents an opportunity to streamline this task.

ResumeGPT2 was created to leverage these technologies in a user-friendly way. It is a web-based application that allows users to upload a resume and a job description in PDF format. The tool then analyzes keyword overlap using natural language processing and constructs a structured prompt for GPT-4 to generate a customized, professional cover letter. The goal of ResumeGPT2 is to reduce the effort required to craft individualized letters while maintaining quality, alignment, and personalization.

2. Problem Statement

The cover letter remains one of the most time-intensive parts of job applications. Manually tailoring each letter requires attention to tone, content relevance, and company-specific language. Many applicants either skip writing letters entirely or rely on generic templates that fail to impress recruiters or make effective use of keywords. Commercial AI-based writing tools often require subscriptions, limit user control, and obscure their decision-making process.

ResumeGPT2 was developed to solve these issues by offering a free, transparent, and customizable tool for cover letter generation. The system not only generates letters based on uploaded content but also allows users to specify tone, traits, and goals, adding a layer of personal branding. By focusing on clear keyword extraction, secure deployment, and structured prompting, the tool balances usability, speed, and effectiveness.

3. Methodology

ResumeGPT2 was built using a modular architecture. The frontend was developed with Streamlit to provide a simple, interactive web interface where users can upload their resume and

job description, as well as input optional personalization fields such as tone and professional goals. Once the user submits the inputs, the backend processes begin.

PDF files are parsed using PyMuPDF, which extracts raw text content from the resume and job description. This text is then processed with spaCy, a natural language processing library, to extract relevant keywords—specifically nouns and proper nouns. These keywords represent areas of potential overlap between the applicant's background and the job requirements. The system then assembles a prompt using a template that includes an introduction, qualifications summary, motivation for applying, and closing. This prompt is passed to the OpenAI GPT-4 API to generate a formatted letter.

Throughout development, security and usability were prioritized. Initially, API keys were hardcoded or entered manually, which was not sustainable or secure. This was addressed by implementing Streamlit Secrets, which allowed for key storage without exposing sensitive information. Prompt formatting was also refined significantly. Early outputs from GPT-4 were inconsistent and disorganized; these issues were resolved by enforcing a structured layout in the prompt that mimicked a standard business letter format.

4. Key Features

The final version of ResumeGPT2 includes a range of features that work together to provide a smooth and effective user experience. Users can upload their resume and job description as PDF files. The system identifies shared language between the two documents, using spaCy to extract relevant nouns that are likely to represent key skills, industries, and responsibilities. Based on this analysis, the app constructs a custom GPT-4 prompt that incorporates the extracted keywords as well as any user-provided tone, traits, or goals.

Once the letter is generated, the output is displayed in the app and can be downloaded in plain text format. The entire workflow is secure and does not require any user login or account creation. Thanks to the use of Streamlit Secrets, the application handles API calls internally without revealing configuration details to the user. The app is hosted live and functions reliably under typical usage conditions.

5. Challenges and Solutions

Several technical and usability challenges arose during development. One of the first issues encountered was instability in GPT-4 outputs. Without clear formatting in the prompt, the language model produced letters that varied significantly in quality and structure. This was addressed by implementing a paragraph-based prompt template with clear headers and consistent tone, which led to more stable and readable outputs.

Keyword extraction initially produced noisy or irrelevant results. Common but unhelpful terms were frequently included, cluttering the prompt and reducing effectiveness. By filtering for

proper nouns and lemmatizing results in spaCy, the system became much better at isolating meaningful keywords that actually reflected job-resume alignment.

Another significant improvement involved deployment. Early iterations required users to manually input API keys or modify code to run the app locally. Moving to Streamlit Secrets allowed for secure deployment without exposing keys or needing local configuration. Finally, the user experience was simplified by consolidating steps into a single page workflow, minimizing friction during use.

6. Results and Evaluation

ResumeGPT2 was tested with several sample resumes and job descriptions, including a representative use case involving a job seeker with experience in sustainability and data analysis applying for a clean energy analyst role. The system successfully identified overlapping terms such as “energy,” “data,” and “analysis,” and generated a professional cover letter that emphasized those themes. The letter also incorporated the user’s specified tone and goal, resulting in a cohesive and personalized output.

In general, user feedback highlighted the speed and clarity of the tool. Letters were generated in under 10 seconds, and the output was typically well-structured and job-relevant. The app performed reliably in multiple test scenarios, and users reported that the ability to customize tone and focus areas made the experience feel more personalized than other AI-based writing tools they had tried.

7. Future Work

There are several planned enhancements for ResumeGPT2. A key improvement is batch processing, which would allow users to generate letters for multiple jobs using different resumes and job descriptions. Another goal is to support downloadable PDF output with professional formatting, enabling users to submit letters directly with job applications.

Longer-term ideas include creating an application dashboard where users can track which jobs they’ve applied to, manage previously generated letters, and receive alerts for resume-job mismatches. Finally, while GPT-4 currently powers the system, future versions may support local models like those available via Ollama, reducing dependency on external APIs and improving cost efficiency.

8. Conclusion

ResumeGPT2 demonstrates how generative AI and natural language processing can be used to automate and improve the process of writing cover letters. By combining file parsing, keyword matching, and GPT-4-based text generation in a secure, streamlined application, the tool offers a real solution to a widespread user need.

The project reflects core competencies developed in EGR 404, including prompt engineering, modular software design, NLP integration, and secure deployment. It is functional, extensible, and offers a foundation for future work in generative assistant systems.

Project Timeline

April 20 – Resume/job parsing MVP
April 22 – Keyword matching with spaCy
April 25 – Prompt customization interface
April 27 – Secure API deployment
April 28 – Documentation and GitHub setup
May 3 – Final video and slides prepared
May 5 – Final submission