SimuATS Final Report

https://github.com/jamesdfurtado/simu-ats/tree/main

1. Introduction

In today's job market, most companies use Applicant Tracking Systems (ATS) to automatically screen resumes before a human recruiter ever sees them. These systems are designed to pick out relevant skills, qualifications, and keywords to decide whether a resume should move forward in the hiring process. The problem is that job seekers often don't fully understand how these systems work, which makes it difficult to tailor their resumes effectively. This leads to qualified candidates being filtered out simply because their resumes aren't optimized for ATS.

My project, SimuATS, was built to help with this. The goal was to create a tool that simulates how an ATS scanner would evaluate a resume and provide AI-based feedback. Not only does SimuATS score how well a resume matches a job description, but it also suggests ways to improve the resume and even generates a revised version. The entire system is designed to be transparent so users can actually understand why their resume is being rated a certain way.

2. Problem Statement

Most people either tailor their resumes by hand or use expensive resume optimization tools. Manual tailoring is time-consuming and error-prone, while paid tools don't usually explain their decisions or allow much customization. I wanted to solve both of these problems with SimuATS.

The objective was to build a free (or at least low-cost) tool that used artificial intelligence to analyze and improve resumes. It had to be easy to use, explain its scoring logic, and allow the user to directly see which skills were important and how their resume measured up.

3. Methodology

3.1 System Architecture

SimuATS has two main parts: a frontend and a backend. The frontend was built using Streamlit, which allows users to upload their resume and job description files directly into the web app. The backend is a modular Python system with four major components:

- extract_skills.py: extracts important skills and keywords from the job description.
- skill_ranker.py: ranks how important each skill is for the job.
- resume_matcher.py: scores how well the resume covers those skills.
- suggestions.py: offers suggestions for improving the resume's wording.

To rank the importance of skills, I used a pre-trained AI model called BART MNLI. This model can classify text in a "zero-shot" way, meaning it can make judgments about new data without having been specifically trained for that exact task. I chose this because it is free, and trained on observing the semantic meaning of sentences. For resume suggestions and revised resume generation, I used the OpenAI API.

3.2 Development Decisions

Originally, I wanted to avoid any paid APIs and rely only on free, open-source models. However, during development, I found that fine-tuning a language model for skill ranking would take far too long and require resources I didn’t have. Zero-shot classification using BART MNLI was a practical compromise-- it was free and worked reasonably well, though not perfectly.

Later in the project, I decided to use the OpenAI API for generating improved resume suggestions. While this added some cost, it significantly increased the quality and usefulness of the suggestions. I made sure to keep this part modular so that in the future, it can be swapped out for a locally running model if needed.

3.3 Tools & Libraries

I used a variety of tools and libraries to build SimuATS, including:

- Streamlit for the web interface
- NLTK for natural language processing
- HuggingFace Transformers for the BART MNLI model
- PyPDF2 and docx2txt for reading PDF and Word documents
- dotenv for managing API keys and environment variables

4. Key Features

SimuATS includes several important features that work together to help users improve their resumes. First, users can upload both job descriptions and resumes directly into the app. The system then extracts relevant skills from the job description and ranks their importance using AI. After that, it evaluates the resume to see how well it covers those skills and assigns a relevance score. Based on this analysis, the app provides clear, specific suggestions for improving the wording of the resume. Finally, it can also generate a fully revised version of the resume that reflects those suggestions. All of this happens in a transparent way, allowing users to understand exactly why their resume received a certain score and what can be done to improve it.

5. Challenges & Solutions

One major challenge was the volatility of the AI models. The zero-shot classification sometimes gave inconsistent results depending on slight changes in the input. To address this, I created a hard-skills list and designed a more rigid prompting system for BART MNLI to keep the results stable.

Another issue was model computation time. Running the models on my CPU was too slow, so I switched to using my GPU. Even then, the initial load time for the models was long. However, since the final goal is to deploy the website online where it will stay running, this loading time should not affect users once deployed.

Managing dependencies was also a challenge. I cleaned up the requirements.txt file and manually handled GPU-specific package installations to ensure the app runs smoothly across different setups.

6. Results & Evaluation

SimuATS is fully functional. Users can upload their resumes and job descriptions, receive skill importance rankings, see how their resume matches up, and get AI-powered suggestions for improvement. In testing, the system was able to improve sample resumes by increasing their relevance scores after applying the suggestions. While I wasn’t able to conduct a formal user study, I tested the app with several example resumes and job descriptions. The feedback from classmates and early testers indicated that the tool provided useful insights and made the resume tailoring process much easier and clearer.

7. Future Work

There are several upgrades planned for SimuATS. First, I will replace the ChatGPT API with a locally running language model using Ollama to reduce costs. I also plan to replace the BART MNLI zero-shot classifier with a custom-trained model that can better assess skill importance based on the language context.

Additionally, I want to add soft skill extraction and ranking, which is currently missing from the tool. I am also reaching out to recruiters to better understand how real ATS systems work, so I can improve the accuracy of the simulation.

Finally, I plan to fully deploy SimuATS onto my personal website so that anyone can use it without needing to install anything.

8. Conclusion

SimuATS successfully demonstrates how AI can be used to simulate an ATS and provide valuable feedback to job seekers. The project balanced cost, accuracy, and usability, making smart trade-offs where necessary. I gained a deeper understanding of natural language processing, AI model integration, and software architecture while developing this tool. The lessons learned will guide future improvements as I continue developing SimuATS into a fully deployable, publicly available app.

Project Timeline (all hit on time)

Note: the beginning of this timeline is offset because I changed my project halfway through

- April 24: Proposal submission
- April 26: Text-based input and scoring MVP
- April 29: Resume generation & UI feedback components
- May 2l: Integration polish + file upload handling
- May 5: Project demo and final submission