JobPilot: An LLM-Based Agent for Resume Parsing, Form Filling, and Application Tracking

1 Introduction

Job applications place a considerable burden on candidates, not only because of their repetitive procedures but also due to the lack of role-specific customization and limited visibility of application progress. Firstly, applicants are required to repeatedly enter the same personal information into disparate recruitment portals and applicant tracking systems (ATS) such as Workday, Beisen, or BOSS Zhipin's enterprise platform, which wastes time and effort. Secondly, resumes are frequently submitted in a generic form without targeted adjustments for each position, leading to weak alignment. Candidates are expected to emphasize different experiences for distinct roles and to adapt similar positions according to each company's particular business focus. Finally, after submission, application progress is dispersed across emails, text messages, phone calls, recruitment websites, and job-seeking apps. To consolidate information, candidates often rely on manual spreadsheets; however, checking the real-time status of a specific company or position still requires opening links or searching entries, which is an inefficient and cumbersome process.

Current solutions inadequately address these challenges. Robotic process automation (RPA) tools automate form-filling by mimicking clicks/keystrokes but lack semantic understanding (e.g., cannot interpret "relevant project experience" context) and are fragile to interface changes—minor recruitment portal layout tweaks render their scripts useless [1]. Job boards' "one-click apply" only works in their own ecosystems, not supporting company-specific portals with customized interfaces outside mainstream frameworks [2]. Traditional natural language processing (NLP) tools parse resumes/job descriptions (e.g., extract skills/requirements) but lack end-to-end automation (e.g., link parsed data to form filling/submission) and interactive progress management (e.g., update status via user feedback) [3-4]. These limitations reflect broader AI tool trends: most systems excel at single tasks but struggle with semantic comprehension or workflow integration [5-7]. In short, current approaches either lack content interpretation ability (RPA) or complete workflow execution capacity ("one-click apply"/traditional NLP).

Recent advances in LLM-based autonomous agents hold great potential for recruitment workflows. For example, AutoGPT uses memory management and external tool integration to autonomously handle user tasks—this could be adapted for resume parsing and application submission [8]. Similarly, ReAct agents combine reasoning with action execution, supporting dynamic job-resume matching (e.g., LLMs interpret unstructured resumes and map them to job requirements) [9]. Generative Agents research highlights role-playing capabilities, where simulated professional agents could assess candidate-job fit by mimicking recruiter decision-making [10]. These models provide generic LLM-based agent frameworks but have not yet been concretely applied to job seeking.

To address these limitations, we propose JobPilot, an LLM-based Agent for automated job application management. In this system, the LLM serves as the semantic engine: it extracts structured information from resumes, analyzes job descriptions, aligns candidate profiles with form fields, and generates tailored content such as resume revisions or written responses. The agent component operates as the orchestrator: it decomposes tasks, invokes web automation tools to complete applications, updates progress records, and interacts with the user through natural language queries. In combination, the LLM provides semantic understanding and content generation, while the agent ensures task planning and execution, together forming an intelligent assistant that streamlines and manages the entire application process.The proposed system is expected to deliver the following key functionalities:

1) **Resume ingestion and profile construction.** Users upload a PDF or DOC resume, and the agent automatically builds a structured candidate profile for future use.

2) **Automated form filling.** Given a job application link, the agent opens the webpage, recognizes form fields, and fills them with appropriate information. Users only need to handle CAPTCHAs and confirm submission.

3) **Job description analysis and resume optimization.** When a job description (JD) is provided, the agent interprets the requirements, computes a relevance score, and generates suggestions or an optimized resume tailored to the position.

4) **Application progress management.** The system maintains a unified dashboard of application records (time, platform,

status, pending tasks). Users can update progress through natural language input (e.g., *"I completed the assessment for Company X"* or *"My application at Company Y has been closed"*), which the LLM interprets to automatically adjust the records. They can also query the system in natural language (e.g., *"Check the application status at Tencent"*), and the agent retrieves and presents the relevant entries for quick review.

2 Technical Summary

We intend to leverage existing LLM application framework such as LangChain or Langflow with Python language to implement the backend core functionalities. The way to access model service would be either utilizing API calling of existing LLM service providers (ChatGPT, Gemini, Qwen or etc.), or deploying the LLM service locally for demostration purpose. The frontend construction would be based on web development framework such as Vue/React with an interactive fashion. Other techniques showcased in the following functionalities will be attempted to construct and improve the system.

1) **Resume ingestion and profile construction.** For PDF/DOC format resume input, the system can parse it via PyPDF2 and python-docx; The key functionality of profile construction will be implemented by the named entity recognition (NER) capability of LLM. With a list of descriptions of the preference postions provided, a vector database could be constructed to record the positions through sentence or document embedding, and the alternative choices for vector database include Pinecone, MongoDB and etc.

2) **Automated application form filling.** After constructing the candidate profile, users may require the system to autmatically fill in the corresponding application form on the website. The system will generate several tool callings or function callings, such as utilizing Playwright or Selenium to do the automated work. To efficiently indentify the keywords of different application forms, it might be necessary to use visual recogniztion and visual understanding abilities of multimodal LLM. To match different keywords with the candidate profile, again it relies on the NER capability of LLM.

3) **Position analysis and resume optimization.** This module will leverage LLM for deep semantic analysis. When a user provides a job description (JD), the system first utilizes the LLM's information extraction capabilities to parse the text, identifying and structuring key information such as required skills, years of experience, and preferred qualifications. Subsequently, this structured data is vectorized through sentence embedding models such as Sentence-BERT. The system calculates a relevance score by computing the cosine similarity between the JD vector and pre-stored candidate resume vectors. For resume optimization, a Retrieval-Augmented Generation (RAG) approach is employed. The system retrieves the most relevant experiences from candidate resumes based on the JD's key requirements. These experiences, along with the JD itself, are fed as contextual information to the LLM. Finally, the LLM generates a tailored resume draft for the position based on the prompt, optimizing project descriptions and highlighting specific skills to maximize job fit.

4) **Application progress management.** The system will utilize a traditional database (e.g., SQLite) to store all application records (such as Company, Position, Application Status, etc.). The entire process is driven by an LLM serving as a Natural Language Understanding (NLU) engine: when users input natural language commands, the LLM parses their intent and entities. When users need to update a status, for example, by typing "My application process at Company A has concluded," the system first performs intent recognition (determining "update application status") and entity extraction (identifying [Company: Company A] and [Status: Process concluded]). Subsequently, the agent converts this structured information into specific database commands (e.g., UPDATE applications SET status = 'closed' WHERE company = 'Company A') and executes them. When users query information, such as asking "Check the application status for Company C," the LLM converts this natural language question into a structured database query. After the agent executes the query and retrieves relevant records, the LLM presents the results to the user in a clear, conversational format.

Team Members

Name	Stu_id	E-mail
LI JIAHUA	3036654405	u3665440@connect.hku.hk
CAI ZIJIAN	3036653865	u3665386@connect.hku.hk
LI RUIWEN	3036653578	u3665357@connect.hku.hk
LU ZHENGHONG	3036656764	u3665676@connect.hku.hk
HE HAOFAN	3036655722	u3665572@connect.hku.hk

References

[1] Ichter B, Brohan A, Chebotar Y, et al. Do as I can, not as I say: grounding language in robotic affordances. In: Proceedings of the 6th Conference on Robot Learning. 2023, 287−318

[2] Wang G, Xie Y, Jiang Y, et al. Voyager: an open-ended embodied agent with large language models. 2023, arXiv preprint arXiv: 2305.16291

[3] Shen Y, Song K, Tan X, et al. HuggingGPT: solving AI tasks with chatGPT and its friends in hugging face. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023

[4] Yao S, Zhao J, Yu D, et al. ReAct: synergizing reasoning and acting in language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023

[5] Chen W, Su Y, Zuo J, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. arXiv preprint arXiv:2308.10848

[6] Liu B, Jiang Y, Zhang X, et al. LLM+P: empowering large language models with optimal planning proficiency. 2023, arXiv preprint arXiv: 2304.11477

[7] Karpas E, Abend O, Belinkov Y, et al. MRKL systems: a modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. 2022, arXiv preprint arXiv: 2205.00445

[8] Richards, M., & Rao, V. (2023). AutoGPT: Autonomous GPT with memory and tool integration. arXiv preprint arXiv:2304.03442.

[9] Yao, S., Zhao, J., Yu, D., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. Proceedings of the International Conference on Learning Representations (ICLR).

[10] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive simulacra of human behavior. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI).