Individual Homework 02-Part 1 Report

Agentic AI CV Verification System via MCP (LinkedIn/Facebook)

Course: Agentic AI for Business and FinTech (FTEC5660)
Student: [CHEN Junhao]
Student ID: [1155251265]

1. Overview

This assignment implements an Agentic AI-based CV (resume) verification system. It reads a candidate's resume text and connects to social graph tools via the MCP (Model Context Protocol) to search, match, and cross-verify information from the candidate's LinkedIn/Facebook profiles. The ultimate goal is to output a structured verification report. The system's objective is not simply to "find an account with the same name," but to perform evidence-based judgment using multiple fields such as educational background, work experience, location, and skills, thereby identifying potential discrepancies in the CV.

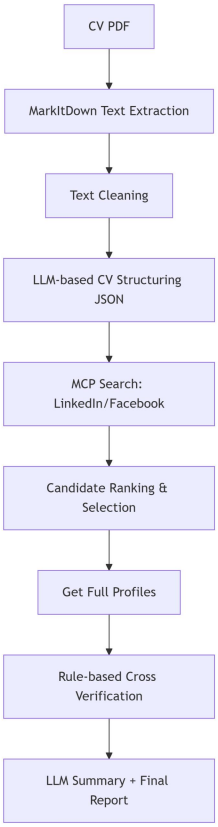
The overall design adopts a hybrid architecture of "LLM + MCP Tools + Rule-based Comparison": the LLM handles unstructured text and tool invocation decisions, MCP tools access social platform information, and the rule layer performs field-level verification and discrepancy classification. This design ensures flexibility while enhancing interpretability, avoiding over-reliance on the model's subjective judgment.

2. System Architecture and Design Decisions

The system consists of five core modules: CV Reading and Cleaning, Structured Information Extraction, Social Profile Candidate Search, Field-level Verification, and Report Generation. First, I used `MarkItDown` to convert the PDF resume into text and performed basic cleaning (removing garbled characters, table separators, redundant spaces, etc.). Then, an LLM is used to extract the resume content into a structured JSON format (name, location, education, work experience, skills, etc.), allowing subsequent verification logic to use a unified data format.

During the social profile verification stage, the system prioritizes LinkedIn (as it contains more comprehensive professional information) and uses Facebook as a supplementary verification source (for location, social connections, personal information, etc.). The candidate search doesn't simply take the first result; instead, it performs candidate scoring and ranking by combining name, location, education, employer, and skills. Subsequently, the system retrieves the full profile of the best candidate and performs cross-verification with the CV at the field level. Finally, the system organizes the discrepancy results into a structured report, providing an overall judgment (verified / partially_verified / suspicious / unable_to_verify) and a confidence score.

Figure 1. System Workflow (Architecture)



Key Design Decisions:
1.Using a structured JSON data model (e.g., `CVProfile`, `ExperienceItem`, `EducationItem`) to reduce complexity in subsequent processing.
2.Adopting a LinkedIn-first, Facebook-auxiliary verification strategy to improve the accuracy of professional information verification.
3.Using a rule layer to output discrepancy evidence (field, CV value, social value, severity), enhancing report interpretability.
4.Distinguishing between "no social evidence found" and "clear conflict discovered" as different statuses to reduce the risk of false positives.

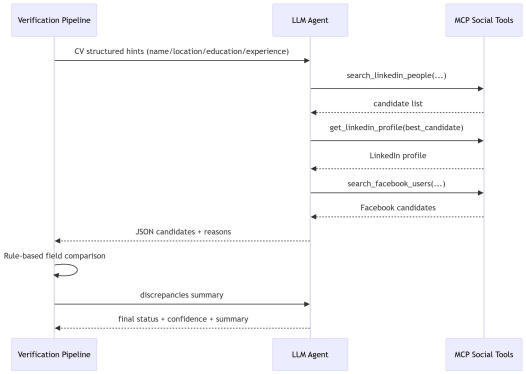
3. Agent Workflow and Tool Usage Strategy

The Agent workflow in this system employs a "tool-augmented workflow" rather than a fully open-ended conversational agent. Specifically, the LLM takes on different responsibilities at each stage: during CV parsing, the LLM extracts structured fields from noisy resume text; during the candidate search stage, the LLM decides how to invoke MCP tools like `search_linkedin_people` and `search_facebook_users` based on resume clues (name, location, school, company, skills); during the result summarization stage, the LLM synthesizes the discrepancy information output by the rule layer into a natural language conclusion.

Regarding the tool usage strategy, I adopted the following process:
(1) First, search for candidates with the same name on LinkedIn.
(2) Re-rank candidates by combining location, school, employer, and job title.

(3) Use `get_linkedin_profile` to retrieve the full professional profile.
(4) Search Facebook to assist in confirming location or identity consistency.
(5) Perform field-level comparison for education and work experience.
(6) Output the verification report and a list of discrepancies.

This strategy aims to maximize the hit rate within a limited number of tool invocation steps while avoiding excessive ineffective searches.
Figure 2. Agent Tool Interaction (Sequence)



4. Verification Logic and Discrepancy Detection

Field-level verification is categorized into three types: Identity Verification, Education Verification, and Work Experience Verification. Identity verification primarily compares name and location; education verification compares school name, degree, major, and graduation year; work experience verification compares company, job title, and start/end dates. Due to potential formatting differences between resumes and social profiles (e.g., "BSc" vs "Bachelor of Science"), the verification results are classified as `match / partial_match / mismatch / missing / unverifiable`, with severity levels introduced as `low / medium / high`.

At the implementation level, I used simple string normalization and similarity matching (e.g., company name alias normalization, keyword matching for job titles) as baseline rules. For example, if the CV claims "ByteDance, Engineer, 2020 – Present" and a matching company and similar job title can be found on LinkedIn, it is judged as `match` or `partial_match`; if both company and job title do not match, it is marked as `mismatch`. For missing social data, the system does not directly judge it as "suspicious" but primarily marks it as `unable_to_verify` to avoid misinterpreting privacy settings or low-activity users as fraudulent resumes.

5. Experimental Results on Sample CVs

I conducted batch tests on 5 sample CVs. The system successfully completed the entire process from PDF reading, information extraction, social search, to report generation. Experimental results show that the system can generate detailed evidence explanations when sufficient LinkedIn clues (name, location, education, employer) are present. However, when public social information is insufficient or candidates are highly common-named, the system typically outputs

`unable_to_verify`. Additionally, the current version exhibits inconsistencies between the "final summary" and the "discrepancy list" (see CV_1 case), indicating that the alignment between the rule layer and profile field mapping still needs further refinement.

Table 1. Summary of Verification Results (Actual Output)

CV File	Detected Person	Overall Status	Confidence	# Discrepancies
CV_1.pdf	John Smith	verified	0.12	3
CV_2.pdf	Minh Pham	unable_to_verify	0.95	4
CV_3.pdf	Wei Zhang	partially_verified	0.47	3
CV_4.pdf	Rahul Sharma	unable_to_verify	0.95	4
CV_5.pdf	Rahul Sharma	unable_to_verify	0.95	5

> Note: The above table is directly taken from the system output (`outputs/summary.json`). The confidence values reflect the model's final scoring result.

6. Case Study (CV_1: John Smith)

The final system output for `CV_1` was verified, with a confidence score of 0.12. During the candidate selection phase, the system identified a high-scoring LinkedIn candidate (candidate_id = 9). The matching reasons included: consistent name (John Smith), consistent location (Singapore), Engineer at ByteDance since 2020, marketing-related Bachelor's background from McGill University (graduated 2009), and high overlap in skills like Content Creation / SEO / Social Media with the CV. The system also retained a Facebook candidate (candidate_id = 213) as an auxiliary candidate, but due to mismatched job (Scientist at Traveloka) and educational clues, it was only considered a moderately confident candidate and not used as the primary verification basis.

In the automatically generated natural language summary, the system considered LinkedIn candidate 9 to provide strong support for the CV's identity, work, education, location, and skill information. It also noted that other candidates with the same name had more obvious conflicts in education, employer, or industry. Therefore, it judged the CV information as "generally accurate with supporting online evidence." This indicates that the candidate search and profile selection modules performed well in this case.

Table 2. Selected Profiles and Interpretation for CV_1

Platform	Candidate ID	Score	Interpretation
LinkedIn	9	0.96	Strong match: same name, ByteDance Engineer (2020–present), Singapore, McGill BSc (Marketing), overlapping skills
Facebook	213	0.55	Moderate but uncertain match: same name and Singapore/HK linkage, but job and education signals conflict

7. Limitations and Future Improvements

This system still has several limitations. Firstly, PDF resume parsing may suffer from issues like broken tables, garbled text, or disordered fields, leading to missing or incorrect extraction during the LLM phase. Secondly, the public availability of social platform information varies greatly; some candidates may have no usable LinkedIn/Facebook information, or their profiles may be

incomplete, affecting the system's verifiability. Furthermore, the common-name problem significantly increases matching difficulty in the absence of auxiliary information like location, school, or employer.

Future improvements can be pursued in three directions: (1) Stronger candidate re-ranking strategies (e.g., incorporating embedding similarity or timeline consistency scoring); (2) More granular date parsing and alignment (standardizing comparisons like "May 2020" vs "2020"); (3) Multi-evidence fusion and human review interfaces (e.g., automatically aggregating high-risk items into a manual checklist). These enhancements could further improve the system's practicality and robustness in real-world recruitment/KYC scenarios.

8. Reproducibility and Repository Contents

The code and report for this project have been organized and are available in a public GitHub repository, containing the following:

`report.pdf`: This report (⩽5 pages)

`main.ipynb` / `main.py`: Main process code

`src/`: Modules for CV reading, structured extraction, MCP tool invocation, candidate matching, verification, and report generation

`outputs/`: Structured results and verification reports for sample CVs (JSON / Markdown)

The execution flow is: Read CV PDF → Text Cleaning → LLM extracts structured resume → MCP searches LinkedIn/Facebook → Retrieve profile → Field-level verification → Output final report.