Agentic CV Verification via MCP SocialGraph

This report describes a CV verification system that extracts claims from PDF CVs and cross-checks them against LinkedIn and Facebook-style profiles provided through the SocialGraph MCP (Model Context Protocol) server. The system outputs a reliability score in $[0, 1]$ for each CV and writes a structured JSON verification report for inspection.

1 System Design

The system is implemented as a small pipeline with three responsibilities: (1) parse a CV PDF and extract structured fields, (2) retrieve social profiles by calling MCP tools, and (3) compare CV claims to social data to detect discrepancies and compute a final score. The implementation is lightweight (pure Python with `requests` and `PyPDF2`) to make it easy to run locally and to keep the decision logic transparent.

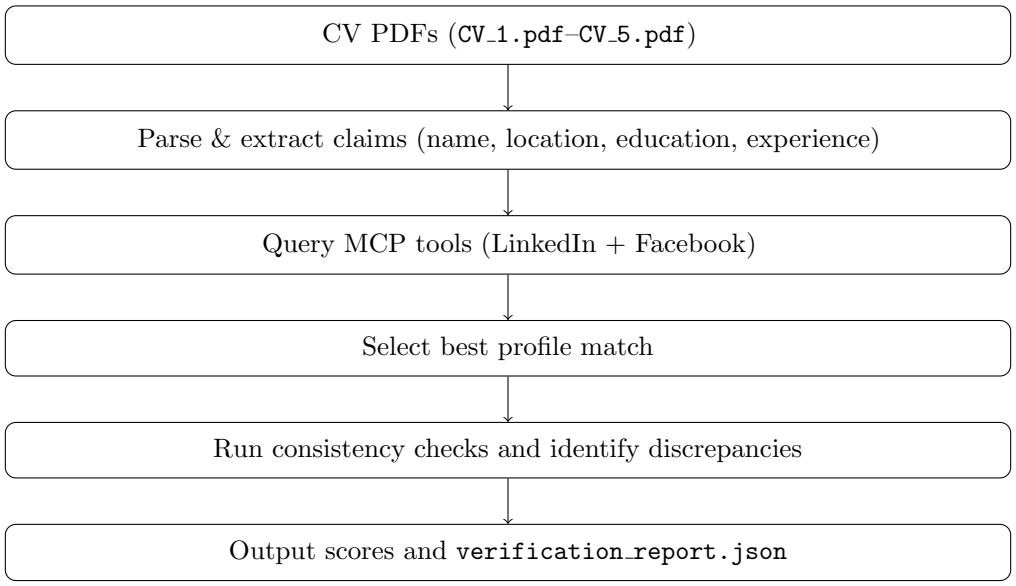

Figure 1: End-to-end workflow for one CV.

Two practical decisions improve robustness. First, PDF text is normalized to reduce layout noise, and location extraction is conservative to avoid treating decorative icons or sentence fragments as places. Second, the MCP endpoint returns SSE (server-sent events), so the client performs an `initialize` handshake, stores the `mcp-session-id`, and parses `data: {...}` lines to recover JSON-RPC responses.

2 Agent Workflow and MCP Tool Usage

For each CV, the agent queries both networks and then selects the best match. On LinkedIn it calls `search_linkedin_people` using the extracted name and an optional location hint, then retrieves the top-ranked candidate via `get_linkedin_profile`. On Facebook it calls `search_facebook_users` and retrieves the best candidate via `get_facebook_profile`. When available, `get_linkedin_interactions` is used as a weak signal for activity and engagement.

Candidate ranking is based on similarity between CV and profile fields (name similarity, location similarity, and keyword overlap with the LinkedIn headline). This makes the workflow resilient to fuzzy matches and minor profile variations because it evaluates multiple candidates instead of assuming the first hit is correct.

3 Discrepancy Checks and Scoring

The system returns a score in $[0, 1]$ for each CV. Scoring starts from a base confidence and subtracts penalties when inconsistencies are detected. The checks focus on interpretable red flags: future-dated education or employment years (e.g., an end year later than the current year), overlapping employment spans, low coverage between CV-listed companies/schools and LinkedIn history, and mismatches between CV employer claims and the Facebook `current_company` field when present. Location mismatch is detected by comparing the CV location set against the social profile location and penalizing when no location matches sufficiently.

The final score is clipped to $[0, 1]$. In evaluation, a threshold of 0.5 is used: scores above 0.5 are treated as reliable; otherwise the CV is flagged for review.

4 Sample Results and Reproducibility

The system writes `verification_report.json` (full extracted fields, selected matches, issue codes) and prints the required list of five scores. Running on the provided five sample CVs produces the following summary.

CV	Candidate Name	Score	Flags (summary)
1	John Smith	0.56	LinkedIn education coverage, Facebook company mismatch
2	Minh Pham	0.76	Facebook company mismatch
3	Wei Zhang	0.84	None
4	Rahul Sharma	0.00	Future year (2027), LinkedIn exp/edu low coverage
5	Rahul Sharma	0.38	LinkedIn exp low coverage, overlap, Facebook company mismatch

Table 1: Verification outputs on sample CV PDFs. Decision threshold is 0.5.

With the 0.5 threshold, the decisions are CV1–CV3 as reliable and CV4–CV5 as problematic. CV4 is penalized strongly because it contains a future end year (2027), which is a clear timeline violation. CV5 is flagged due to overlapping experience spans and weak alignment with LinkedIn experience. In general, results should be interpreted as probabilistic evidence because social profiles can be incomplete and PDF extraction can be noisy.