AI-Powered Profile Consolidation using Local SLMs and Agentic Workflows

1. Abstract

Maintaining consistent professional profiles across platforms like resumes and LinkedIn is a persistent challenge for individuals, often leading to inaccuracies that can impact career opportunities. This paper introduces an AI-powered system designed for automated extraction, comparison, and consolidation of professional data. The system leverages local Small Language Models (SLMs) within an agentic framework, specifically Microsoft AutoGen, orchestrated through a Directed Acyclic Graph (DiGraph) of specialized agents. Core to its methodology are meticulous prompt engineering, the enforcement of structured output via Pydantic models, and the application of few-shot learning techniques to enhance agent reliability. Evaluation on consumer-grade hardware demonstrates the viability of local SLMs for such complex tasks, with models like qwen3:30b-a3b achieving 00% accuracy in data extraction, and Deepseek-r1:14b presenting a strong balance of performance and accuracy. The primary contributions include a functional offline profile consolidation system, empirical insights into the performance of local SLMs in agentic workflows, and a demonstration that deliberate, programmatic design of agent interactions can be more effective than relying on unguided Large Language Model capabilities.

2. Introduction

Professionals today navigate a complex digital landscape where maintaining a consistent and accurate representation of their skills and experiences across multiple platforms is crucial. Resumes and LinkedIn profiles serve as primary conduits for career advancement, yet discrepancies between them are common. These inconsistencies can arise from infrequent updates, differing formatting requirements, or simply the oversight inherent in manually managing multiple versions of one's professional narrative. The consequences can be significant, ranging from missed job opportunities due to outdated information to a diminished professional image. The manual effort required to synchronize these profiles is not only time-consuming and tedious but also highly susceptible to human error. For instance, educational qualifications might be listed with varying degrees of detail: one resume version might comprehensively list a Master's degree with expected graduation date, GPA, and relevant projects, while another version or a LinkedIn profile might present a condensed summary, perhaps omitting GPA or specific project details, or adding different contextual information like associated skills or extracurricular activities.

To address these challenges, this paper presents an "AI-Powered Profile Consolidation" system. This system offers a novel solution by employing an agentic AI pipeline built using the Microsoft AutoGen framework and powered by locally deployed Small Language Models (SLMs). Its core function is to parse user-provided PDF resumes and static PDF representations of LinkedIn profiles, extract relevant information, compare corresponding entries, and generate a single, consolidated JSON output. A key characteristic of the system is its entirely offline operation, which ensures user data privacy and provides full control over the process, distinguishing it from cloud-based solutions. The development journey prioritized accessibility, aiming for a system that could be run effectively on standard consumer hardware without incurring API costs, thereby making advanced AI tools more widely available.

The main contributions of this work are:

1. The design and implementation of a multi-agent system utilizing a Directed Acyclic Graph (DiGraph) architecture. This structured approach enables robust and modular processing for profile data extraction and consolidation from multiple sources.
2. A demonstration of the efficacy of local, open-source SLMs, typically in the 2-4 billion parameter range, for performing complex, structured information extraction and reasoning tasks. This is achieved on consumer-grade hardware, highlighting the increasing capabilities of smaller models.
3. A comprehensive evaluation of various SLMs, introducing a custom performance score. This score is designed to balance extraction accuracy and processing speed, with a particular emphasis on penalizing inaccuracies, which is critical for high-stakes information tasks like profile management.
4. Valuable insights into effective prompt engineering strategies for enhancing LLM reliability within agentic workflows. This includes the use of Pydantic models for enforcing structured outputs and the application of few-shot learning to guide model behavior effectively.

The remainder of this paper is structured as follows: Section 3 reviews related work in resume parsing, agentic AI, and SLM advancements. Section 4 details the system architecture and the agentic workflow. Section 5 describes the implementation specifics, including data preprocessing and prompt engineering. Section 6 presents the evaluation methodology and results. Section 7 discusses the findings, challenges, and limitations. Finally, Section 8 concludes the paper and outlines future research directions.

3. Related Work

The task of profile consolidation intersects several active research and development areas, including information extraction from documents, web data retrieval, agentic AI systems, and the application of modern language models.

Resume Parsing and Information Extraction:
Numerous commercial services (e.g., Sovren, Textkernel, Jobright.ai) and open-source tools exist for parsing resumes and extracting structured information. Many open-source solutions leverage NLP libraries like spaCy. While these tools can effectively extract entities such as education, work experience, and skills from individual documents, they often struggle with the wide variety of resume formats encountered in practice and typically do not support cross-platform reconciliation of information from disparate sources like LinkedIn. Their primary focus is often employer-facing, geared towards candidate screening rather than empowering individuals to manage their own data consistency.

Web Data Extraction and Agentic AI Frameworks:
Conventional web scraping tools such as Playwright and Selenium are commonly used for extracting data from websites. However, these tools can be brittle, relying on hardcoded selectors that break with website updates, and are often easily detected and blocked by platforms like LinkedIn, especially when performing actions that require login. Agentic AI frameworks like Microsoft AutoGen , LangChain , and CrewAI offer a more sophisticated approach, enabling the creation of autonomous agents that can perform complex tasks, including web navigation and data processing. While these frameworks hold significant promise, their specific application to the nuanced task of reconciling professional profile data from sources like resumes and LinkedIn, particularly with an emphasis on using local models for privacy and cost-efficiency, remains an underexplored area.

Small Language Models (SLMs) and Efficient AI:
Recent advancements in Small Language Models (SLMs) have demonstrated their growing capability to perform complex reasoning and generation tasks, often approaching the performance of much larger models but with significantly reduced computational requirements. Several techniques contribute to the efficiency and viability of SLMs for local deployment:

- **Mixture-of-Experts (MoE):** MoE models, such as Alibaba's Qwen3 MoE, activate only a subset of "expert" parameters per input token. This dynamic routing optimizes for both speed and specialized reasoning, reducing overall computational load.
- **Compact Instruction-Tuned Models:** SLMs like Meta's Llama 3.2 (B and 3B instruct versions) are fine-tuned on instruction datasets to follow user prompts effectively, making them well-suited for agentic tasks. These models can achieve strong performance in dialogue and instruction-following tasks despite their small size (often GB - 5GB).
- **Quantization:** Techniques like quantizing model weights from 6-bit floating point

to 8-bit integers (or even lower) significantly compress model size and accelerate inference with often minimal loss in accuracy. Google's gemma3 and various Deepseek R models offer quantized versions (e.g., it-qat for "instruction-tuned quantization-aware training").
- **Distillation:** This process involves training a smaller "student" model to mimic the behavior of a larger, more capable "teacher" model. Distillation can transfer the performance benefits of large models to smaller, more efficient ones. Deepseek R1, for example, offers models distilled from larger architectures. These advancements are crucial as they make it feasible to deploy sophisticated AI systems like the one presented in this paper on consumer-grade hardware, aligning with the project's goal of accessibility and privacy.

Data Reconciliation and Semantic Similarity:
Traditional approaches to data reconciliation often rely on rule-based systems or machine learning techniques for entity resolution. While effective for structured data, these methods may lack the nuance required for comparing text-heavy profile sections. The initial proposal for this project considered using embedding similarity techniques with models like Sentence-BERT to quantify similarity between text fields, alongside LLM-based semantic analysis. The final system, however, leans more heavily on the integrated reasoning capabilities of LLM agents within the AutoGen framework to perform the comparison and merging, guided by carefully crafted prompts. This shift reflects an evolution towards leveraging the holistic understanding of LLMs for the consolidation task itself.
Positioning the Current Work:
The novelty of this project lies in the synergistic integration of an agentic AI framework (Microsoft AutoGen) with locally deployed SLMs to achieve end-to-end professional profile consolidation. Unlike systems that rely on cloud APIs or basic string matching, this work emphasizes a structured agent interaction model (the DiGraph), meticulous prompt engineering to ensure reliable and formatted output, and offline operation for enhanced user privacy and control. It provides empirical evidence for the effectiveness of modern SLMs in a practical, multi-step information processing task on readily available hardware.

4. System Architecture and Approach

The design of the AI-Powered Profile Consolidation system is rooted in a core philosophy: **Deliberate Agentic Design**. This principle emphasizes the creation of a structured, programmatic workflow for agent interactions, rather than relying on the unguided, and sometimes unpredictable, capabilities of raw LLMs. This approach promotes reliability, maintainability, and predictability in the system's output. The development was guided by the ethos of making AI tools more accessible, aiming for a system that students or individuals could run on their personal computers free of cost and with reasonable processing times. This philosophy directly addresses lessons learned from earlier, less successful attempts with overly complex frameworks or misused agent capabilities, where a lack of explicit control led to debugging overhead and inconsistent results.

Overall Pipeline Orchestration (final_test.py):
The main entry point and orchestrator of the entire multi-agent workflow is the final_test.py script. This script is responsible for:

1. Initializing clients for various open-source LLMs (such as Qwen, Llama, Gemma) accessed via the Ollama service, which facilitates local model deployment.
2. Defining a suite of specialized agents using the Microsoft AutoGen framework. Each agent is designed for a specific sub-task within the consolidation pipeline.
3. Constructing a directed graph to manage and dictate the flow of information and control between these agents.

Agentic Graph (AutoGen DiGraph):
A key architectural component is the use of AutoGen's DiGraphBuilder to create a Directed Acyclic Graph (DAG) that governs agent interactions. This is not a simple linear chain of agents but a structured workflow that explicitly defines which agent passes information to which other agent, and under what conditions. This graph-based structure provides a clear and manageable way to orchestrate complex multi-agent conversations, offering a balance between the flexibility of agent-based systems and the determinism of programmatic logic. This design choice was a direct response to the challenges encountered with earlier approaches, such as the perceived over-engineering of LangChain/LangGraph for this specific task or the initial underutilization of AutoGen's multi-agent potential in a single-program flow. The DiGraph allows for modularity, where each agent focuses on its expertise, and their collaboration is explicitly mapped.
A conceptual representation of this agentic graph is shown below :

```
+-------------+     +----------------------+     +-----------------------+
+------------------+
| User Input |----->| Input Agent (Edu/Work)|----->| Resume_Edu_Extractor |----->| |
| (Resume PDF,| +----------------------+     +-----------------------+ |
Combine_Edu Agent |-----> Consolidated
| LinkedIn PDF)| | LinkedIn_Edu_Extractor |----->| | JSON (Edu)
+-------------+                    +-----------------------+
+------------------+
|
| +----------------------+     +--------------------+
            +--------------------------------->| Resume_Work_Extractor |----->| |
                                  +------------------------+ | Combine_Work Agent
|-----> Consolidated
| LinkedIn_Work_Extractor|----->| (Partially Impl.) | JSON (Work)
| (Partially Impl.) | +---------------------+
                                  +------------------------+
```

Figure : Conceptual Diagram of the Agentic Workflow. Arrows indicate the flow of data and control between agents. Parallel extraction paths for Education and Work Experience are shown, followed by respective combination agents.

Parallel Processing Flows:
The agentic graph is designed with two primary parallel processing flows, one for education information and one for work experience, which can run concurrently up to the combination stage. This parallelism is a deliberate design choice to improve efficiency by allowing independent extraction tasks to proceed simultaneously.

- **Education Flow (Fully Implemented):** This pipeline is complete and operational. It begins with an Input agent that receives the initial trigger. This agent then activates two Extractor agents in parallel: one dedicated to extracting education details from the resume (Resume_Edu_Extractor) and another for extracting education details from the LinkedIn profile (LinkedIn_Edu_Extractor). The structured outputs from both these extractor agents are then passed to a Combine_Edu agent. This agent is tasked with intelligently merging the data, resolving discrepancies, and producing a single, comprehensive JSON object representing the user's consolidated education history.
- **Work Experience Flow (Partially Implemented):** This flow is designed to mirror the structure and logic of the Education flow but is currently only partially implemented. The system includes an agent for extracting work experience from the resume (Resume_Exp_Extractor). The agents responsible for extracting work experience from LinkedIn (LinkedIn_Work_Extractor) and the subsequent Combine_Work agent are defined in the codebase but are commented out. This indicates that their full implementation and integration are planned next steps in the project's development. This transparency regarding the current state of implementation is important for accurately representing the system's capabilities.

Execution Trigger:
The entire multi-agent consolidation process is initiated by sending a single, simple message, "Start the flow," to the initial agent in the graph. This triggers the cascade of agent conversations and actions defined by the DiGraph. The system streams the entire process, including intermediate agent messages (for debugging and transparency) and the final

consolidated JSON output, directly to the console.

5. Implementation Details

The practical realization of the AI-Powered Profile Consolidation system involved careful attention to data handling, sophisticated prompt engineering, and local LLM deployment. These implementation choices were critical for achieving reliable and accurate performance.

Data Handling and Preprocessing (/data, /common):
The system processes two main types of input data: user resumes and LinkedIn profiles, both provided in PDF format.

- **Input Data:** The /data folder serves as the repository for raw input. It contains subfolders for resumes (e.g., self.pdf, self2.pdf in /data/resumes/) and a PDF version of a user's LinkedIn profile (e.g., self.pdf in /data/lkd/). The use of a static PDF representation for the LinkedIn profile was a key requirement from the initial project proposal and successfully met the "Minimal Promised Outcome". This decision also pragmatically sidestepped the complexities and risks of live web scraping, such as potential IP bans, which were encountered in early exploratory phases.
- **PDF to Markdown Conversion:** A crucial preprocessing step is handled by the constants.py script located in the /common folder. This script utilizes the docling library to convert the raw PDF documents into clean, structured Markdown text. This Markdown version, not the original PDF, is what the LLM agents receive as input. This choice aligns with the lesson learned to "use programming libraries instead of LLMs when possible" for deterministic tasks like basic document parsing , as it simplifies the input for the LLMs and allows them to focus on their core strength: understanding and extracting information from text, rather than dealing with complex PDF structures.
- **Model Configurations:** The constants.py file also centralizes the configurations for all the different open-source LLMs used in the project. This makes it straightforward to swap models for different agents or for evaluation purposes, facilitating experimentation and iterative development.

Prompt Engineering (/prompts): The "Brain" of the Project
The /prompts folder houses the meticulously crafted prompts that define each agent's behavior and instructions, effectively acting as the "brain" of the system. The success of the LLM agents hinges on the quality and specificity of these prompts.

- **Structured Output Definition (out_ext.py):** To ensure consistency and facilitate downstream processing, the system mandates a strict JSON schema for the outputs of all extractor agents. This is achieved using Pydantic models (e.g., OutExtEdu for education extraction, OutExtExp for experience extraction) defined in out_ext.py. These models specify the expected fields, data types, and structure of the JSON output. This enforcement guarantees that every agent, regardless of the underlying LLM, returns data in the exact same format, which is essential for the Combine agents to reliably merge the information. The initial design of these Pydantic models drew inspiration from the more comprehensive data structures envisioned in the early Basic_Planned_Functionality.pdf document, which outlined ideal fields for various profile sections like education and work experience.

- **Few-Shot Examples (fewshot_ext.py):** Recognizing that LLMs often benefit from concrete examples, fewshot_ext.py provides high-quality few-shot examples for each extraction task. For instance, for the task "extract education from resume," the file contains an example of an input resume text snippet and the corresponding, perfectly formatted JSON output that the agent is expected to produce. These examples are dynamically injected into the main system messages for the agents. This technique was found to dramatically improve the LLMs' accuracy and their adherence to the extraction rules and output schema. This directly addresses the empirical finding that "single-shot prompting failed for nuanced resume/LinkedIn tasks," necessitating the use of few-shot examples for robust performance.
- **Master System Messages (sysmsg_ext.py):** The sysmsg_ext.py script is responsible for constructing the master system message for each agent. These are not static instruction sets but are dynamically assembled to be comprehensive and context-specific. Each system message typically includes:
  1. A clear definition of the agent's core role and objective.
  2. The full pre-processed Markdown text of the resume or LinkedIn profile relevant to the agent's task.
  3. A list of critical instructions and strict rules for extraction (e.g., specific date formatting requirements, how to handle null or missing values, what to do with ambiguous entries).
  4. The relevant few-shot examples from fewshot_ext.py to guide the LLM's output generation. The agent names themselves are designed to be mnemonic to enhance clarity and maintainability (e.g., sysmsgExtResEdu clearly indicates it's the system message for the agent that Extracts Education information from a Resume). The combination of Pydantic models for structural enforcement and few-shot examples for content and style guidance creates a powerful mechanism for controlling LLM behavior, leading to more reliable and predictable structured data extraction.

LLM Deployment (Ollama):

The system utilizes Ollama for the local deployment and management of various open-source SLMs. This allows for easy experimentation with different models like Qwen, Llama, Gemma, and Mistral. The project primarily focused on models in the 2-4 billion parameter range, which generally require under 0GB of RAM. This choice ensures that the system remains accessible and can be run effectively on consumer-grade hardware, including high-end laptops, without incurring API costs or raising data privacy concerns associated with cloud-based LLM services.

6. Evaluation

The evaluation of the AI-Powered Profile Consolidation system focused on assessing the performance of various locally deployed SLMs in the context of the defined agentic workflow. The primary goals were to measure accuracy in information extraction and processing speed, and to understand the trade-offs involved.

Experimental Setup:
All evaluation tests were conducted on an M Max 32GB MacBook Pro. This hardware represents a high-end consumer laptop, demonstrating the feasibility of running the described complex agentic workflows locally. The input data for evaluation consisted of sample resumes (self.pdf, self2.pdf) and a static PDF representation of a LinkedIn profile (self.pdf), as described in the /data directory structure. The system processed these inputs through the Education extraction pipeline, which is fully implemented.
Evaluation Metrics:
To quantify performance, the following metrics were used:

- **Accuracy (%):** This was determined by comparing the LLM-generated JSON output against a manually annotated ground truth. The presentation materials indicate "Error Fields (Out Of /30)" , implying 30 key fields were targeted for extraction. Accuracy is calculated as ((30 - Error Fields) / 30) * 00.
- **Inference Duration (s):** This measures the total time taken for the entire consolidation process for a given model, from the initial "Start the flow" message to the generation of the final JSON output for the Education section.
- **Custom Performance Score:** A custom metric, the Performance Score, was designed to provide a holistic measure that balances accuracy and speed, with a strong emphasis on correctness. It is calculated as Performance Score = ((Accuracy/100)^3 * 100) / Duration. The rationale behind cubing the normalized accuracy term (Accuracy / 00) is to heavily penalize lower accuracy scores. For tasks like profile data management, where errors can have significant negative consequences, high accuracy is paramount. A model with 90% accuracy is thus penalized much more significantly relative to a 00% accurate model than it would be with a linear accuracy term. For example, an accuracy of 60% (0.6) results in an effective accuracy component of 0.63=0.26 in the numerator, while 90% (0.9) accuracy yields 0.93=0.729. This non-linear penalty reflects the critical

importance of data fidelity.

Models Evaluated:
A range of open-source SLMs, varying in size and architecture, were evaluated. These models were deployed locally via Ollama. The selection included :

- qwen3:30b-a3b (8 GB)
- Deepseek-r1:.5b (. GB)
- Deepseek-r1:4b (9.0 GB)
- gemma3:12b (8. GB)
- gemma3:12b-it-qat (8.9 GB, instruction-tuned and quantization-aware trained)
- llama3.2:3b-instruct-fp6 (6.4 GB)
- Deepseek-r1:32b (9 GB)
- gemma3:4b (3.3 GB)

Quantitative Results:
The performance of the evaluated models is summarized in Table .
Table : Evaluation Results of Various SLMs on Profile Data Extraction (Education Flow)

Model	**Size**	**Duration (s)**	**Error Fields (Out Of /30)**	**Accuracy (%)**	**Performance Score**
qwen3:30b-a3b	8 GB	46.65	0	00	2.4
Deepseek-r1:.5b	. GB	2.46	2	60	.73
Deepseek-r1:4b	9.0 GB	68.06	2	93	.8
gemma3:12b	8. GB	04.49	0	00	0.96
gemma3:12b-it-qat	8.9 GB	05.7	0	00	0.95
llama3.2:3b-instruct-fp6	6.4 GB	38.22	2	60	0.57
Deepseek-r1:32b	9 GB	37.9	3	90	0.23

gemma3:4b	3.3 GB	25.47	22	27	0.08

(Data sourced from)

Analysis of Results:
The evaluation results reveal several important findings:

- **High Accuracy Achievable:** Several models demonstrated perfect or near-perfect accuracy. Notably, qwen3:30b-a3b, gemma3:12b, and gemma3:12b-it-qat all achieved 00% accuracy (0 error fields) on the tested dataset. This confirms that local SLMs, when appropriately prompted within a structured agentic system, can perform high-fidelity information extraction.
- **Performance Trade-offs:** The qwen3:30b-a3b model not only achieved 00% accuracy but also did so with a significantly faster inference time (46.65s) compared to the gemma3:12b models (around 05s), resulting in the highest Performance Score (2.4). This highlights the trade-off between model architecture, size, and efficiency.
- **Balanced Performers:** The Deepseek-r1:4b model, while not achieving perfect accuracy (93%, 2 error fields), offered a compelling balance. Its Performance Score of .8 was competitive, suggesting it could be a strong candidate where a slight compromise in absolute accuracy is acceptable for potentially faster processing or lower resource usage compared to the top performers, although its duration was higher than qwen3:30b-a3b in this test. The final presentation noted it as having the "best overall performance" , which might consider factors beyond this specific table, or a different interpretation of "overall."
- **Impact of Inaccuracy:** Models with lower accuracy, such as gemma3:4b (27% accuracy) and llama3.2:3b-instruct-fp6 (60% accuracy), received significantly lower Performance Scores, demonstrating the effectiveness of the custom metric in penalizing errors heavily. This underscores the importance of selecting models that can meet high accuracy thresholds for this application.
- The evaluation approach, while focusing on overall field accuracy and a composite score, represents a pragmatic initial assessment. The initial project proposal envisioned more granular metrics like Precision, Recall, and F-score for different information categories. The shift to the current metrics likely reflects practical constraints, with more detailed evaluation planned as future work.

Meeting Project Goals:
The successful extraction and consolidation of education data with high accuracy by several models demonstrate that the system met and exceeded the "Minimal Promised Outcome" outlined in the project proposal. This outcome required extracting data from PDF resumes and

static LinkedIn representations, performing a comparison, and identifying discrepancies for key fields like Education and Work Experience. The fully implemented Education flow validates this achievement.

7. Discussion

The development and evaluation of the AI-Powered Profile Consolidation system yield several important findings and highlight practical considerations for building agentic AI systems with local SLMs.

Key Findings & Implications:

1. **Viability of Local SLMs for Complex Agentic Tasks:** This project provides strong evidence that complex agentic workflows, involving multiple steps of information extraction, comparison, and structured generation, can be run effectively on consumer-grade hardware using locally deployed, open-source SLMs. Models with 2-30 billion parameters demonstrated high accuracy, and even smaller models showed promise. This has significant implications for making sophisticated AI tools more accessible and affordable, removing reliance on expensive proprietary APIs and enhancing user data privacy by keeping all processing local.
2. **Critical Performance Trade-offs in Model Selection:** The evaluation underscored the crucial trade-offs between model accuracy, inference speed, and resource requirements (model size, RAM usage). While gemma3:12b achieved perfect accuracy, its speed was less than qwen3:30b-a3b, which also achieved perfect accuracy and thus a better performance score. The Deepseek-r1:4b model, with 93% accuracy, was noted as delivering good overall performance, suggesting that for some applications, a slight compromise on absolute accuracy might be acceptable if it brings benefits in speed or resource footprint. This emphasizes that there is no single "best" model; selection must be tailored to the specific requirements and constraints of the task and deployment environment.
3. **Deliberate Design Outweighs Raw LLM Capability:** A central finding is that the reliability and success of the system hinged more on deliberate, programmatic design choices than on the raw, unguided intelligence of the LLMs. Key design elements contributing to this were:
    - The structured agent interaction defined by the AutoGen DiGraph.
    - Rigorous prompt engineering, including clear role definition, provision of full context, strict rules, and few-shot examples.
    - Preprocessing of input PDFs into clean Markdown using dedicated libraries (docling), simplifying the task for the LLMs.
    - Enforcement of structured JSON output via Pydantic models. This approach

of programming deterministic logic where possible and reserving LLM capabilities for tasks requiring nuanced understanding (like semantic interpretation and data merging) proved more effective and reliable than attempting to have LLMs manage the entire workflow with minimal guidance.

Challenges Encountered & Lessons Learned:
The project journey was not without its challenges, which provided valuable lessons:

- **Agentic Framework Complexity:** Navigating the landscape of agentic AI libraries like AutoGen and LangChain revealed a lack of standardization and varying levels of maturity. This led to a significant setup time and occasional inconsistent behaviors that required careful debugging. The choice of AutoGen with a DiGraph was a deliberate attempt to find a balance between power and manageable complexity for this project's scale.
- **Resource Limitations:** Operating entirely on local, consumer-grade hardware, even high-end, imposed constraints, particularly regarding memory capacity when experimenting with LLMs larger than 0-5 billion parameters. This reinforced the focus on optimizing for efficient SLMs.
- **Limited Open-Source Stack Support:** The open-source agent ecosystem is still evolving. There was a lack of built-in support for some essential tasks, such as robust PDF parsing (addressed by integrating docling) or seamless management of structured memory between agent turns (partially addressed by explicit context passing and Pydantic models).
- **Context Management Between Agents:** Ensuring that each agent in the DiGraph had the precise context it needed, while isolating it from irrelevant information from previous steps, was a design challenge. "Learned context" or outputs from one agent had to be thoughtfully structured (Pydantic models were crucial here) and explicitly passed to the next, requiring careful design of these intermediary data structures.
- **Web Navigation Risks (Informing Design Choices):** An early exploration into dynamic LinkedIn scraping using Playwright led to an incident of accidentally sending too many concurrent requests, nearly resulting in an IP ban. This experience served as a strong cautionary tale and directly motivated the decision to use static PDF representations of LinkedIn profiles for the current implementation. This pragmatic shift mitigated risks and allowed focus on the core consolidation logic, deferring robust and ethical live web navigation to future work. It highlights how real-world constraints can, and should, shape system design.

The "Failed Approaches" documented in the project's evolution were instrumental in shaping the final, successful architecture:

- **Full LinkedIn Scraping with Playwright:** Deemed too brittle and prone to breaking with UI changes.
- **LangChain/LangGraph:** Considered too complex and over-engineered for the relatively linear (though multi-step) nature of profile processing tasks in this project. The debugging and orchestration overhead was high.
- **AutoGen-core (Initial Attempt):** Initially used in a single-program flow, which defeated its multi-agent potential and lacked flexibility for modular task design. These experiences reinforced key lessons: agent frameworks should be modular yet simple enough to manage; programming libraries are preferable to LLMs for deterministic tasks like PDF parsing; prompt engineering is a time-intensive but critical component; and dynamic, unguided agent interactions should be minimized where logic can be explicitly programmed or prompted to reduce randomness and improve reliability.

Limitations of the Current Study:
This study, while demonstrating significant progress, has several limitations:

- **Scope of Consolidation:** The system currently provides a fully implemented pipeline for the "Education" section of a professional profile. The "Work Experience" section is only partially implemented (resume extraction is complete, LinkedIn extraction and combination are pending). Other potentially valuable sections outlined in initial planning documents, such as Projects, Skills, and Certifications , are not yet incorporated.
- **Data Diversity and Generalizability:** The evaluation was conducted using a limited set of sample resumes and a single LinkedIn profile representation. While performance was strong on this dataset, the system's generalizability to a wider variety of resume formats, writing styles, industries, and LinkedIn profile structures needs more extensive testing.
- **Static LinkedIn Representation:** The current system relies on a static PDF export of the LinkedIn profile. It does not perform live, dynamic scraping of LinkedIn data, which would be necessary for handling real-time updates and accessing the most current information.

8. Conclusion and Future Work

Conclusion:
This paper has presented an AI-Powered Profile Consolidation system that successfully leverages local Small Language Models within a structured agentic framework (Microsoft AutoGen) to automate the extraction, comparison, and merging of professional data from resumes and LinkedIn profiles. The project demonstrates the viability of running complex,

multi-agent AI workflows on consumer-grade hardware, offering a pathway to more accessible, private, and affordable AI solutions. Key findings underscore the impressive capabilities of modern SLMs for nuanced information processing tasks, the critical performance trade-offs inherent in model selection, and, crucially, the paramount importance of deliberate, programmatic design in building reliable and effective agentic systems. By prioritizing structured agent interactions and meticulous prompt engineering over unguided LLM autonomy, the system achieves high accuracy and provides a valuable tool for individuals seeking to maintain consistent and accurate professional profiles.

Future Work:
The current system provides a strong foundation for several exciting avenues of future research and development:

- **Robust Evaluation Pipeline:** Establish a more formal and comprehensive evaluation framework. This would involve creating a larger, more diverse benchmark dataset of resumes and LinkedIn profiles, and implementing finer-grained metrics (e.g., precision, recall, F-score per field type) to better understand model performance on specific extraction sub-tasks.
- **Increase Data Diversity:** Systematically expand testing across a wider range of resume formats (e.g., chronological, functional, academic CVs), industries, languages, and LinkedIn profile variations. This will be crucial for improving the system's generalizability and its ability to handle edge cases robustly.
- **Complete Work Experience Flow:** Prioritize the full implementation of the Work Experience pipeline, including the LinkedIn experience extraction agent and the corresponding combination agent, to mirror the functionality of the Education flow.
- **Expand Scope to Other Profile Sections:** Extend the system's capabilities to parse and consolidate other important profile sections, such as Projects, Skills, Professional Certifications, Publications, and Related Coursework, as initially envisioned. This would involve designing new Pydantic models, prompts, and potentially specialized agents for each new section.
- **Safe and Ethical Web Navigation by Agents:** Explore the development of fully autonomous web agents capable of reliably and ethically logging into platforms like LinkedIn, navigating to user profiles, and extracting data in real-time. This would require addressing challenges related to dynamic web content, anti-scraping measures, and user consent, building upon the lessons learned from initial web navigation attempts.
- **End-to-End System for Career Management:** Envision a more cohesive, end-to-end system that not only consolidates profiles but also leverages this consolidated data to assist users with other career-related tasks, such as identifying relevant job postings or even navigating browser-based application

forms
- **Live User Interaction and Refinement:** Implement a smooth interactive loop where users can review the consolidated profile, approve or reject suggested changes, provide feedback to the agents, and make decisions in real-time. This would transform the system into a collaborative tool, enhancing user trust and control

Addressing these future work directions will further enhance the utility and robustness of the AI-Powered Profile Consolidation system, moving closer to the goal of a comprehensive AI assistant for professional self-presentation and career development.

9. References

- Wu, Q. et al. (2023). Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155.
- Reimers, N., & Gurevych, I. (209). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.
- Microsoft Playwright. Playwright Documentation. (Available: https://playwright.dev/)
- Ollama. (Available: https://ollama.com/)
- Docling Project. (Available: https://github.com/docling-project)
- Pydantic. (Available: https://pydantic-docs.helpmanual.io/)