Evaluating Meaning-Typed Programming for Data-Intensive LLM-Based Information Extraction

Kristen Vinh
Rochester Institute of Technology
Golisano College of Computing and I.S.
kv1895@rit.edu

Zikun Xu
Rochester Institute of Technology
Golisano College of Computing and I.S.
aax2388@rit.edu

02/06/2026

1 Introduction

Large Language Models (LLMs) are increasingly integrated into production software for tasks such as information extraction, data cleaning, and the processing of semi-structured content [1]. In practice, most AI-integrated systems interact with LLMs through manually-crafted prompts and writing of application-specific post-processing code. Because the use of manually-written prompts often implicitly encode task logic and constraints, changes in requirements often require updates to both prompts and parsing logic, complicating maintenance and contributing to fragile system behavior as applications become more complex [2, 3].

Meaning-Typed Programming (MTP) introduces an alternative model for integrating LLMs into software systems by shifting the specification of intent from natural language prompts to the program itself. Rather than explicitly constructing prompts, developers express desired behavior through typed function signatures and native language constructs, assigning execution to an LLM using the `by <model>` operator. Semantic information extracted by the compiler and enforced at runtime is used to automatically synthesize model inputs and convert model outputs into typed program values[4].

This paper investigates whether MTP reduces engineering complexity in data-intensive information extraction pipelines, such as extracting data from job listing websites, compared to traditional prompt-engineered implementations.

2 Background and Related Work

Most AI-integrated applications that rely on LLMs are built using prompt-engineered workflows, in which developers manually construct natural language prompts and then are required to process model outputs to create structured data [1, 2]. While this approach enables rapid experimentation and lowers the barrier to entry for non-expert users[7], it introduces software engineering challenges related to maintainability and evolution. Frameworks such as DSPy and LMQL attempt to add structure to prompt-based interaction, but still require developers to perform explicit prompt formulation and output handling [8]. Prior studies of these frameworks emphasize model accuracy, cost, and runtime performance, with comparatively limited attention to development effort and code complexity [3].

Meaning-Typed Programming (MTP) shifts task specification from natural language prompts to typed program constructs [4]. Using MTP, developers are able to automate prompt engineering with semantically rich, human readable code expressions, while prompt synthesis and output handling are delegated to compiler and runtime mechanisms. Initial evaluations of MTP report reductions in lines of code and development time, and therefore, further investigation of its applicability in practical, data-intensive settings should be conducted [4].

Web scraping represents a practical application for evaluating the use of MTP for integrating LLMs into software systems. A common use of web scraping is the aggregation and collection of job listings from multiple online platforms [5, 6]. The aggregation of job listings has a range of applications, including helping users efficiently identify job opportunities that align with their skills and educational background on various job portals, as well as allowing analysis of labor market trends and the skills or educational qualifications required within a given field [6].

However, job listing websites pose challenges for traditional rule-based scrapers. Key attributes such as salary, required skills, and educational level are often embedded in unstructured text blocks or expressed using inconsistent formats (e.g., hourly, daily, or annual compensation) [6]. These characteristics demonstrate the need for web scraping approaches that provide greater flexibility in extracting semi-structured information while maintain-

ing manageable implementation complexity. This makes job listings a useful test case for assessing whether MTP-based implementations can reduce development effort and code complexity while maintaining accurate data extraction compared to traditional AI-integrated methods.

3 Problem Definition and Research Question

Many AI-integrated approaches for web scraping require prompt engineering, structuring inputs, and optimizing wording and context. In addition, other frameworks for integrating LLMs into programming, such as DSPy and LMQL, tend to incur additional complexity and have somewhat steep learning curves [4]. This research aims to identify if a new framework, MTP, can reduce lines of code (LOC), development time, and code complexity compared to traditional AI-web scraping (such as using BeautifulSoup and an LLM for manual prompting, as used in [7]) when collecting data from various job listing websites.

The study is guided by the following research question:

How does the development time, complexity, and lines of code (LOC) for an MTP-based scraper compare to a traditional BeautifulSoup + OpenAI API approach?

4 Dataset Strategy

To determine whether an MTP script can reduce LOC, development time, and code complexity compared to a more traditional BeautifulSoup and OpenAI API approach for web scraping, researchers will build web scrapers to retrieve job information from various job opening platform sites such as Jooble, We Work Remotely, or Wellfound (formerly AngelList), as these sites were identified as likely to be legal for web scraping [9]. As referenced above, job listing websites can often be difficult to parse by more basic scrapers–they often store valuable data like salary, skills, education requirements, etc. in large blocks of text or in different formats (such as listing pay by hour, day or salary) within the same website. Therefore, there is a need to develop web scraping scripts that offer more flexibility to scrape information while avoiding overly complex scripts, making it a good candidate for testing MTP for web scraping.

5 High-Level Architecture and Proposed Solution

This study aims to build two functionally equivalent web scraping pipelines to extract basic job listing information such as salary, skills, education level, and location:

- **Baseline Implementation**: HTML parsing with BeautifulSoup and manual prompt construction using the OpenAI API for data extraction.
- **MTP Implementation**: HTML parsing with BeautifulSoup, followed by using the `by <model>`operator provided by the byLLM Framework for data extraction.

BeautifulSoup is used in both pipelines as a preprocessing step to reduce AI token usage and to keep both processes as comparable as possible, although other methods may be explored [10]. These pipelines will be used to extract data from at least two job listing websites in order to gather multiple sets of code and data to evaluate. Comparisons of the two implementations will focus on software quality metrics such as development time, lines of code, and complexity in addition to accuracy. In order to compare development time researchers will track coding time using a plugin such as WakaTime [9, 11].

Other metrics, including lines of code and complexity, will be analyzed using Python libraries such as radon [9, 12]. While this research is focused on comparing these software quality metrics of these two implementations, the researchers will compare the accuracy of both scrapers (BeautifulSoup + OpenAI vs MTP) by comparing the LLM-extracted data to a set of ground truth data extracted manually (20-30 job listings) [9].

References

[1] T. B. Brown et al., "Language Models are Few-Shot Learners," Jul. 22, 2020, arXiv: arXiv:2005.14165. DOI:10.48550/arXiv.2005.14165.

[2] Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting Is Programming: A Query Language for Large Language Models. *Proceedings of the ACM on Programming Languages*, Vol. 7, POPL, pp. 1–33, 2023. DOI: https://doi.org/10.1145/3591300.

[3] Saleema Amershi et al. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, 2019. DOI: https://www.microsoft.com/en-us/research/wp-content/uploads/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf

[4] J. L. Dantanarayana et al., "MTP: A Meaning-Typed Language Abstraction for AI-Integrated Programming," Proc. *ACM Program.* Lang., vol. 9,

no. OOPSLA2, p. 314:1176-314:1204, Oct. 2025, DOI:10.1145/3763092.

[5] A. Kumar, K. Chauhan, and J. K. Grewal, "Web Scraping Job Portals," in *Advancements in Communication and Systems*, Malviya National Institute of Technology (MNIT), Jaipur, A. K. Tripathi, V. Shrivastava, and National Institute of Technology Delhi, Eds., Soft Computing Research Society, 2024, pp. 291–303. DOI:10.56155/978-81-955020-7-3-25.

[6] S. Lunn, J. Zhu, and M. Ross, "Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice," in *2020 IEEE Frontiers in Education Conference (FIE)*, Oct. 2020, pp. 1–9. DOI:10.1109/FIE44824.2020.9274270.

[7] A. Bhardwaj, N. Diwan, and G. Wang, "Beyond BeautifulSoup: Benchmarking LLM-Powered Web Scraping for Everyday Users," Jan. 09, 2026, arXiv: arXiv:2601.06301. doi: 10.48550/arXiv.2601.06301.

[8] O. Khattab et al., "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," Oct. 05, 2023, arXiv: arXiv:2310.03714. DOI:10.48550/arXiv.2310.03714.

[9] Google, "Measuring Scraper Development Time," Gemini Thinking. Accessed: Feb. 02, 2026. [Online]. DOI:https://gemini.google.com/share/52fbb1c6c68b

[10] Google, "Job Scraper Setups: BS4 vs. BYLLM," Gemini Fast. Accessed: Feb. 03, 2026. [Online]. DOI:https://gemini.google.com/share/c0a65831241f

[11] WakaTime, "WakaTime - Dashboards for developers," WakaTime. Accessed: Feb. 02, 2026. [Online]. DOI:https://wakatime.com/

[12] radon: Code Metrics in Python. Python. Accessed: Feb. 02, 2026. [OS Independent]. DOI:https://radon.readthedocs.org/