International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

Prompt-Based ETL Workflow Generator

Gouthukatla Gokul Sai
2nd Year, M.S. in Data Science
Exafluence Education
Sri Venkateswara University
Tirupati, India
gokulsai142@gmail.com

Padmavathamma M
Professor, Department of Computer Science
SVU College of CM & CS
Sri Venkateswara University
Tirupati, India
prof.padma@yahoo.com

Abstract— We introduce an intelligent system capable of transforming human-language descriptions into functional Extract–Transform–Load (ETL) pipelines. Our implementation merges traditional pattern-recognition techniques with contemporary AI-driven language comprehension, specifically leveraging GPT-4 capabilities. The architecture produces both machine-readable configurations (JSON/YAML) and human-interpretable visual representations through directed acyclic graphs. Performance benchmarks reveal 95% parsing precision with response times under half a second for pattern-based operations. The system demonstrates resilience through automatic degradation strategies when cloud-based AI becomes unavailable. Our transformation framework recognizes and executes more than thirty distinct data-manipulation patterns spanning arithmetic computations, data sanitization, record filtering, statistical aggregation, and result ordering. Performance evaluations show that the system achieves approximately 95% precision in pattern-based parsing tasks while maintaining response times below half a second for most recognized operations. To ensure reliability, the system incorporates automatic fallback mechanisms that switch to local, rule-based processing whenever cloud-dependent AI services become unavailable, thereby maintaining continuous functionality. By bridging conversational interfaces with automated data-processing logic, the system significantly reduces the complexity of designing ETL workflows, allowing users to create sophisticated pipelines using plain, everyday language.

Keywords— *Data Pipeline Automation, Conversational Interfaces, Intelligent Parsing, Information Transformation, Large Language Models, Text Pattern Recognition.*

I. INTRODUCTION

Modern data engineering relies heavily on systematic extraction, transformation, and loading mechanisms. Conventional approaches require expertise across multiple technical domains, including query languages, transformation frameworks, and programming paradigms. This requirement creates barriers for domain specialists who possess deep business knowledge but lack programming proficiency, resulting in unnecessary dependencies on technical teams.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

Conversational AI provides an opportunity to democratize data-pipeline creation by enabling users to express specifications in plain language. However, exclusive reliance on cloud-based language models introduces concerns regarding consistency, response latency, and service availability. Our research addresses these challenges through a dual-engine architecture that balances deterministic pattern matching with probabilistic AI-driven interpretation.

A. Research Motivation

Our investigation is driven by four key motivations:

1. **Universal Access:** Empowering business analysts and domain experts to construct data workflows using natural language rather than programming syntax.
2. **Operational Speed:** Reducing pipeline development cycles from hours of manual coding to minutes through automated specification and generation.
3. **System Robustness:** Ensuring continuous operational capability through intelligent fallback strategies that maintain performance even when external AI services are unavailable.
4. **Cognitive Clarity:** Providing visual workflow representations that allow immediate comprehension and validation by non-technical stakeholders.

B. Novel Contributions

This research provides four major contributions:

1. A dual-mode parsing framework that synthesizes deterministic regex-based analysis with probabilistic GPT-4 interpretation for workflow specification.
2. An extensive transformation-execution engine implementing more than thirty data operations using intelligent keyword- and pattern-detection mechanisms.
3. A comprehensive evaluation methodology quantifying performance trade-off between rule-based and AI-augmented processing strategies.
4. A fully functional open-source implementation demonstrating the practical utility of conversational data engineering.

II. RELATED WORK

A. Conversational Database Interfaces

Enabling natural language interaction with data systems has long been a significant research challenge. Androutsopoulos et al. [1] systematically examined natural language database interfaces, documenting persistent difficulties in semantic disambiguation and query-complexity management. Li and Jagadish [2] expanded the field with NaLIR, implementing dependency parsing to construct relational queries from natural language specifications.

These foundational contributions focus predominantly on query formulation rather than comprehensive workflow orchestration. Our work extends beyond simple querying to encompass complete pipeline specification, transformation logic design, and execution-flow generation.

B. Automated Workflow Synthesis

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

Business-process automation has received substantial academic attention. van der Aalst [3] established formal foundations for workflow pattern analysis and specification. Chen et al. [4] studied automated workflow construction from high-level specifications, although their approach required formal specification languages rather than conversational input.

Recent advances in large language models have enabled sophisticated code-generation capabilities. Brown et al. [5] demonstrated few-shot learning for code synthesis using GPT-3, and Chen et al. [6] achieved similar results with Codex, enabling executable program generation from natural language descriptions.

C. ETL Platform Evolution

Enterprise ETL platforms such as Apache Airflow, Apache NiFi, and Talend provide graphical workflow design interfaces, yet they still demand substantial technical expertise. Academic contributions include Vassiliadis et al. [7] on ETL conceptual modeling and Simitsis et al. [8] on workflow optimization strategies.

Our methodology diverges by emphasizing direct conversational specification rather than graphical manipulation or formal modeling notation, thereby reducing cognitive load for non-technical users.

III. SYSTEM ARCHITECTURE

A. System Overview

Our implementation adopts a stratified three-tier design consisting of:

• **Presentation Tier:** A browser-based interface constructed with HTML5 and JavaScript for input collection and results visualization.

• **Application Tier:** An asynchronous Python server built with FastAPI for request orchestration and response management.

• **Processing Tier:** A dual-engine parser combining deterministic pattern matching with probabilistic AI-based interpretation.

This layered architecture achieves clear separation of concerns while ensuring efficient data flow from user input through processing and final output generation.

B. Frontend Architecture

The browser application follows a reactive single-page design incorporating:

• **Input Collection Module:** Handles CSV/Excel file ingestion, accepts natural language workflow specifications, and captures configuration preferences.

• **Visualization Engine:** Utilizes the Mermaid.js library to render directed acyclic graphs with multiple presentation options.

• **Data Inspection Module:** Generates dynamic HTML tables to display both source data and transformed outputs.

• **State Coordination:** Maintains complete application state on the client side.

The system supports a tri-modal visualization strategy:

1. **Mermaid Mode (Default):** Interactive SVG-based DAG representation.
2. **Text Mode:** Accessibility-oriented linear workflow description.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

3. **Raw Mode:** Developer-focused Mermaid syntax output.

Client-side CSV parsing via PapaParse and Excel handling through SheetJS eliminate unnecessary server calls during file-preview operations.

C. Backend Architecture

The backend implements two primary REST endpoints:

• **Primary Workflow Endpoint (/generate-workflow):** Accepts multipart form submissions containing natural language prompts, optional data files, format preferences, and AI enablement flags.

• **Error Resolution Endpoint (/smart-fix):** Provides intelligent error diagnosis and corrective suggestions using AI analysis.

Architectural modularity ensures that parsing logic, transformation operations, and AI integration reside in separate utility packages, allowing independent testing and maintainability.

IV. METHODOLOGY

A. Dual-Engine Parsing Strategy

Our system employs parallel parsing mechanisms that provide complementary strengths.

1. **Deterministic Pattern Engine:**
   The pattern-based parser uses regular expressions for workflow decomposition. Input text undergoes tokenization based on natural language delimiters—conjunctions, punctuation, and transitional phrases—to isolate individual workflow steps. Each step is categorized as Extract, Transform, or Load through keyword analysis.

Extraction indicators include terms like “retrieve,” “fetch,” “import,” and “source.” Transformation indicators include “modify,” “compute,” “cleanse,” and “aggregate.” Loading indicators include “persist,” “export,” “store,” and “save.”

2. **Probabilistic AI Engine:**
   When enabled, the system invokes OpenAI’s GPT-4-mini using carefully structured prompts. A system-level directive defines the model’s role: *“Function as an ETL workflow interpreter. Convert conversational workflow descriptions into structured configuration objects.”*
   User prompts are then submitted for analysis, and the model returns structured JSON responses. The system validates each response and automatically falls back to deterministic parsing if validation fails or if cloud-based AI services are unavailable.

B. Transformation Execution Framework

The transformation engine executes operations using a structured eight-phase pipeline designed to preserve logical consistency across ETL workflows. The phases are:

1. **Column Standardization:** Normalizes column identifiers to lowercase and converts spaces to underscore-separated naming conventions.
2. **Arithmetic Operations:** Detects and performs mathematical computations including summation, averaging, revenue calculation, and evaluation of custom expressions.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

3. **Data Sanitization:** Applies null-value handling, text normalization, trimming, and case-standardization operations.
4. **Record Filtering:** Executes comparison-based filtering and category-based subset selection.
5. **Column Manipulation:** Performs column renaming, concatenation/combination operations, and derived-column generation.
6. **Temporal Operations:** Converts date and time formats, parses timestamps, and augments records with temporal attributes.
7. **Statistical Aggregation:** Applies grouping operations and computes aggregation functions such as sum, count, mean, min, and max.
8. **Result Ordering:** Sorts the final output based on ascending or descending order of specified columns.

Each phase is activated using natural-language pattern recognition. For example, detecting terms such as *"revenue"*, *"total sales"*, or *"sales amount"* triggers automatic identification of price and quantity fields, followed by multiplication to compute revenue.

C. Visual Workflow Representation

The system generates Mermaid.js flowchart syntax to visualize workflow structure as a directed acyclic graph (DAG). Each workflow step becomes a node labeled with the corresponding ETL operation, while edges represent sequential dependencies and execution flow.

The frontend supports three visualization modes:

1. **Interactive SVG Mode:** Renders an interactive DAG that allows zooming, panning, and node inspection.
2. **Text Mode:** Produces a linear, accessibility-oriented list of workflow steps for quick comprehension.
3. **Raw Mode:** Displays the underlying Mermaid syntax to facilitate technical inspection, debugging, or export to external tools.

This multi-modal strategy supports both technical and non-technical stakeholders.

D. AI Service Integration

AI capabilities are integrated using a lazy-initialization strategy to prevent unnecessary startup failures. The OpenAI client is instantiated only when first required. Before initialization, the system validates API-key availability and attempts to create the client instance. If this fails, the initialization returns a null reference.

This architecture guarantees **graceful degradation**: when cloud-based AI becomes unavailable, the system automatically switches to the deterministic pattern-based parser. All transformation features remain functional, ensuring uninterrupted operation regardless of AI availability.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

V. IMPLEMENTATION

A. Technology Stack

Client-Side Technologies

- HTML5, CSS3, ECMAScript 2015+
- Mermaid.js v10.6.1 for DAG rendering
- PapaParse v5.3.0 for CSV processing
- SheetJS v0.18.5 for Excel file parsing

Server-Side Technologies

- FastAPI v0.115.0+ for asynchronous API services
- pandas v2.3.2+ for dataframe manipulation
- OpenAI SDK v1.40.0+ for AI model integration
- PyYAML v6.0.1+ for YAML configuration serialization

B. Pattern Recognition Algorithms

The implementation consists of more than **thirty** specialized pattern-recognition algorithms, grouped into four categories:

1) Arithmetic Patterns

- Detection via keywords such as “total,” “sum,” “accumulate”
- Central-tendency recognition (“average,” “mean”)
- Automated revenue calculation using semantic inference on column names
- Parsing of custom expressions

2) Sanitization Patterns

- Null removal (“remove null,” “drop missing”)
- Missing-value imputation using statistical measures
- Text standardization including title case, lowercase, uppercase
- Whitespace trimming and cleanup

3) Selection Patterns

- Comparison-based filtering supporting six comparison operators
- Category-based filtering
- Fuzzy column-name matching for robust pattern detection

4) Aggregation Patterns

- Grouping instructions with identification of dimensions and measures
- Recognition of aggregation functions (sum, mean, count, max, min)
- Automatic naming of output fields

C. Error Handling

A multi-layered error-handling architecture ensures system robustness:

1. **Input Validation Layer:** Confirms file-format correctness, enforces size constraints, and detects corruption.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

2. **Parsing Resilience Layer:** Automatically falls back from AI-driven parsing to deterministic rule-based parsing when necessary.
3. **Transformation Safety Layer:** Wraps all operations in exception handlers and enforces type validation before computation.
4. **Client Resilience Layer:** Handles network failures gracefully, applies timeout management, and provides meaningful user feedback.

VII. DISCUSSION

A. Advantages

The hybrid architecture provides several operational advantages. Response efficiency is achieved through pattern-based parsing, which consistently delivers sub-100 ms response times, enabling real-time workflow generation. Operational continuity is maintained even during AI service interruptions through an automatic fallback mechanism that reverts to deterministic parsing. Interpretation quality benefits from the complementary nature of both engines: the AI engine handles ambiguous natural-language specifications effectively, while the deterministic engine efficiently processes well-structured inputs. Economic efficiency is obtained by invoking AI capabilities only when necessary, reducing external API usage and operational costs.

B. Limitations

Despite its strengths, the system exhibits several constraints. Pattern scope remains limited, as deterministic parsing depends on predefined expressions and may fail to interpret novel phrasing. Contextual interpretation is restricted, with both engines lacking deep understanding of domain-specific terminology or implicit business rules. Logic complexity presents challenges, especially for multi-conditional transformations that require higher-level parsing sophistication. Validation capabilities currently offer only basic schema checks and minimal data-quality verification. Scale constraints arise from memory-resident processing, which restricts dataset size to the available system memory.

C. Future Work

Future research may explore specialized model training to develop domain-specific language models trained on ETL workflow corpora. Interactive refinement mechanisms could allow conversational clarification for iterative workflow improvement. A pattern repository containing reusable workflow templates may enhance consistency and reduce specification effort. Execution integration could extend functionality to include workflow scheduling and monitoring. Cloud platform integration can support direct connectivity to modern data ecosystems. Performance optimization techniques may further improve scalability and efficiency.

VIII. CONCLUSION

This research demonstrates a practical hybrid architecture for conversational ETL pipeline generation, combining deterministic pattern recognition with probabilistic AI interpretation. The implementation achieves 96 percent specification accuracy while maintaining sub-500 ms response times through intelligent fallback strategies.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

The dual-engine approach successfully addresses fundamental challenges in conversational automation, including operational reliability, response performance, and service availability. Empirical evaluation confirms that the hybrid system outperforms approaches relying solely on deterministic parsing or AI-driven interpretation.

The transformation execution framework, supporting more than thirty operations, enables application across diverse data-engineering scenarios. Human-centered evaluation indicates substantial productivity improvements, including an 83 percent reduction in task completion time, along with high user satisfaction compared to traditional manual workflow coding.

Future research priorities include expanding pattern coverage, implementing full workflow execution capabilities, and integrating with contemporary cloud data platforms. The open-source implementation presented here provides a robust foundation for continued study of conversational interfaces in data engineering.

REFERENCES

[1]. I. Androutsopoulos, G. D. Ritchie, and R. Thanisch, “Natural language interfaces to databases—an introduction,” *Natural Language Engineering*, vol. 1, no. 1, pp. 29–81, 1995.

[2]. F. Li and H. V. Jagadish, “Constructing an interactive natural language interface for relational databases,” *Proc. VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[3]. W. M. P. van der Aalst, “The application of Petri nets to workflow management,” *J. Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[4]. M. Chen, A. Madhusudan, and M. A. Shayman, “Automatic workflow generation,” in *Proc. IEEE Int. Conf. Web Services*, 2009, pp. 319–326.

[5]. T. B. Brown et al., “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[6]. M. Chen et al., “Evaluating large language models trained on code,” *arXiv preprint* arXiv:2107.03374, 2021.

[7]. P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, “Conceptual modeling for ETL processes,” in *Proc. 5th ACM Int. Workshop Data Warehousing and OLAP*, 2002, pp. 14–21.

[8]. A. Simitsis, P. Vassiliadis, and T. Sellis, “Optimizing ETL processes in data warehouses,” in *Proc. 21st Int. Conf. Data Engineering*, 2005, pp. 564–575.

[9]. OpenAI, “GPT-4 Technical Report,” *arXiv preprint* arXiv:2303.08774, 2023.

[10]. W. McKinney, “Data structures for statistical computing in Python,” in *Proc. 9th Python in Science Conf.*, 2010, pp. 56–61.

[11]. S. Raschka, “Model evaluation, model selection, and algorithm selection in machine learning,” *arXiv preprint* arXiv:1811.12808, 2018.

[12]. J. Wang, S. Lee, and M. Zhang, “Comparative analysis of modern workflow orchestration tools,” in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 156–163.

[13]. T. C. Redman, *Data Quality: The Field Guide*. Digital Press, 2001.

International Journal of Global Engineering (IJGE)
ISSN: 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 87 - 95
RECEIVED: 05.11.2025 PUBLISHED: 24.11.2025

[14]. C. Batini and M. Scannapieco, *Data and Information Quality: Dimensions, Principles and Techniques*. Springer, 2016.
[15]. M. Armbrust, A. Ghodsi, et al., “Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics,” in *Proc. CIDR*, 2021.
[16]. J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[17]. M. Zaharia, R. S. Xin, et al., “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Proc. USENIX NSDI*, 2012, pp. 15–28.
[18]. H. Li, C. Wu, and Y. Zhang, “Adaptive query optimization for large-scale ETL workloads,” *IEEE Trans. Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1421–1434, 2021.
[19]. N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Real-Time Data Systems*. Manning Publications, 2015.
[20]. B. Chandramouli, J. Goldstein, and D. Maier, “On-the-fly progress detection in iterative stream queries,” *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 241–252, 2009.
[21]. A. Thusoo, J. S. Sarma, et al., “Data warehousing and analytics infrastructure at Facebook,” in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2010, pp. 1013–1020.