TutorBot: Architecture Approaches on Pipeline Analysis and Conversational AI Tutoring

Benjamin Euto, Shahaan Khan, Vaibhav Deokar

IST 736 – Text Mining Final Project

Syracuse University

Table of Contents

Abstract

The following analysis and report describes an integrated system for personalized writing assessment and improvement. **TutorBot** employs a four-stage hybrid pipeline for batch analysis of ChatGPT conversation histories, using keyword-based heuristic filtering, batched LLM confirmation, detailed evaluation scoring, and pattern synthesis to achieve approximately 90% cost reduction compared to individual message evaluation. This system demonstrate that AI-driven writing assessment can operate at both macro (historical pattern analysis) and micro (real-time correction) scales, with the pipeline approach providing diagnostic baselines and longitudinal tracking while the agent approach delivers proactive, personalized tutoring experiences.

Introduction

The integration of AI into education has accelerated dramatically with ChatGPT's release. We see AI being used in all factors of life and how we interact with it changes daily. We believe AI will be the best teacher for students as they continue to grow as, just like a cellphone, it will always be in the hands of people. This adoption generates valuable but largely untapped data: conversational logs documenting AI interactions that reveal patterns in writing quality, grammatical challenges, and communication habits.

ESL (English as a Second Language) learners face particular challenges in academic settings due to fundamental differences between their native language (L1) and English (L2). Common difficulties include article usage, verb tenses, subject-verb agreement, preposition usage, and sentence structure. Traditional tutoring methods are often reactive, requiring students to explicitly ask for help, an approach that misses opportunities for embedded, contextual learning. They cannot give students the individual support they need and can cost an extremely large sum of money.

This paper presents a singular approach which addressing both retrospective analysis and active feedback, **TutorBot.** The pipeline-based system for batch analysis of ChatGPT conversation exports, providing diagnostic baselines and longitudinal progress tracking through efficient hybrid processing.

This paper proves that ChatGPT logs are viable for a personalized writing assessment, a replicable cost-effective framework for conversation analysis, an agent architecture for proactive grammar tutoring with L1-interference detection, and a unified vision showing how pipeline and agent approaches complement each other in educational AI systems.

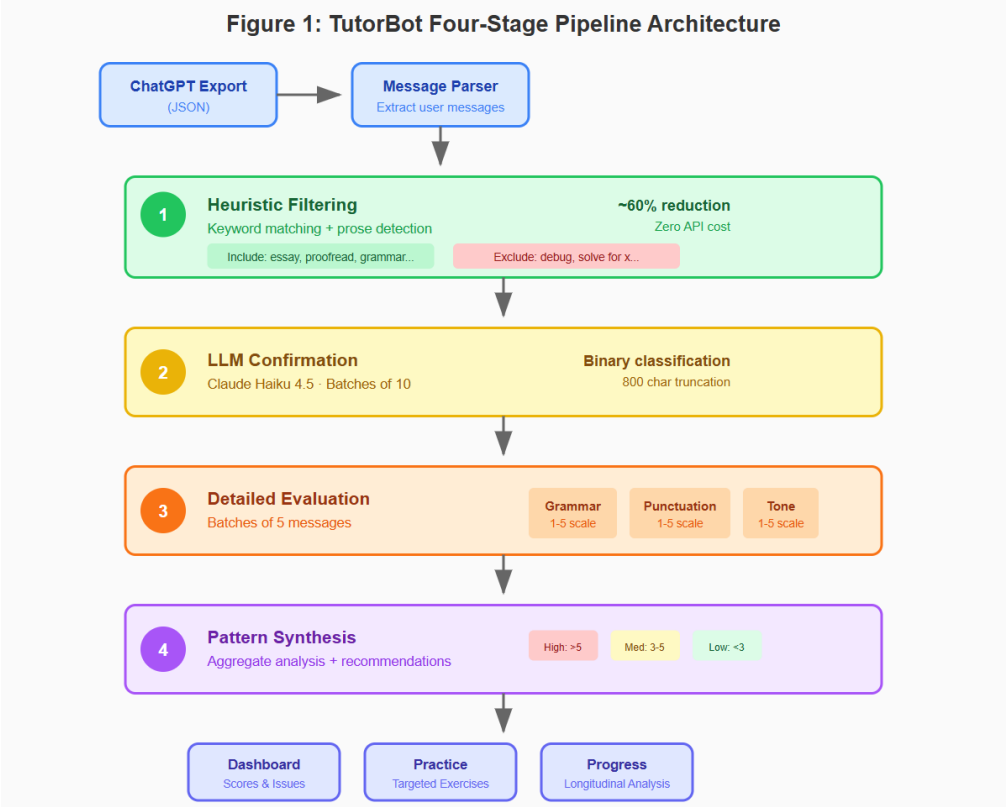
Method

Data Source

This system process ChatGPT conversation exports in JSON format. Users export their data from OpenAI's data export feature, which provides a structured file containing all conversations with nested message mappings. Our parser extracts user messages by traversing the mapping structure, filtering for messages where and extracting text from the array. Each message is tagged with a unique ID, timestamp, conversation index, and conversation title.

Pipeline Architecture (TutorBot)

TutorBot implements a four-stage hybrid pipeline designed to balance classification accuracy with computational efficiency (see Figure 1).



Stage	Process	Reduction	API Cost
1	**Heuristic Filtering** Keyword matching + prose detection	40-60%	**Zero**
2	**LLM Confirmation** Claude Haiku · Batches of 10 · 800 char	20-40%	Low
3	**Detailed Evaluation** Grammar, Punctuation, Tone (1-5 scale)	—	Moderate
4	**Pattern Synthesis** Aggregate + severity + recommendations	—	Low

Table 1: TutorBot Four-Stage Pipeline Summary

Stage 1: Heuristic Filtering

Messages are filtered using two keyword lists. The inclusion list contains 50+ writing-related terms (e.g., 'essay', 'proofread', 'grammar', 'thesis', 'email', 'cover letter'). The exclusion list contains terms indicating non-writing content (e.g., 'debug this code', 'solve for x', 'workout routine', 'recipe for'). Messages under 10 words are filtered. This permissive strategy minimizes false negatives while achieving approximately 40-60% volume reduction with zero API cost. Refer to Appendix B for a full list.

Stage 2: LLM Confirmation

Surviving messages are sent to Claude Haiku 4.5 in batches of 10 for binary relevance classification. Each message is truncated to 600-800 characters to reduce token costs. The

prompt instructs the model to return a JSON array of boolean values, defaulting to true when uncertain to preserve recall. A 60-second rate limit pause occurs every 4 batches to respect API limits. We limited it to 4 instead of 5 as per the free API tier to ensure we did not reach any batch limits. This stage further filters 20-40% of remaining messages.

Stage 3: Detailed Evaluation

Confirmed messages are evaluated in batches of 5. For each message, the model provides grammar score (1-5), punctuation score (1-5), tone score (1-5), and arrays of specific issues per category. Results are mapped back to original message IDs to maintain traceability. Failed batches default to neutral scores (3) with error flags. These are based on the prompt found in Appendix C.

Stage 4: Pattern Synthesis

All evaluations are aggregated and sent to the model for pattern identification. The prompt requires 1-5 issues per category, ordered by frequency, with severity classifications (high: >5 occurrences, medium: 3-5, low: <3) and actionable recommendations.

Technology Stack

Component	TutorBot
Frontend	React 18 + TypeScript + Tailwind
Backend	Client-side (browser)
Database	None (stateless)
LLM	Claude Haiku 4.5
Grammar	LLM-based

Table 2: Technology Stack Comparison

Results

Pipeline Performance (TutorBot)

The four-stage pipeline achieves significant efficiency gains. Stage 1 heuristic filtering typically reduces message volume by 40-60% with no API cost. Stage 2 confirmation further filters 20-40% of remaining messages. The combined architecture reduces API costs by approximately 90% compared to individual message evaluation. Total processing time for typical ChatGPT histories ranges from 1-20 minutes depending on volume and rate limiting pauses.

The web application provides a five-step workflow; it starts with API key validation, then valuation with conversation range selection and progress tracking, then practice sessions with real-time grading, before providing a dashboard displaying scores, issue breakdowns, and session history, and finally, a re-evaluation comparing baseline and follow-up analyses.

Practice Generation and Grading

This system generate practice sessions targeting identified weaknesses. TutorBot provides questions in three formats: correction tasks, multiple choice, and writing prompts. Grading uses a three-tier approach; first, it performs programmatic matching for multiple choice, then does similarity scoring for short answers (threshold: 0.3), and finally performs LLM-based

evaluation for complex responses. The system tracks performance by issue type to identify strengths (more than 80% accuracy) and persistent weaknesses (less than 60% accuracy).

Ethics Statement

Data Privacy: TutorBot processes conversation data entirely client-side. Users upload their own exported ChatGPT logs obtained through OpenAI's data export feature. No conversation data is stored on external servers. Sage uses local LanguageTool processing and PostgreSQL storage under user control. Users retain full ownership of their data. This is not to be used on public networks as it is now and only run locally.

Informed Consent: Users voluntarily upload their own conversation history and provide their own API keys, ensuring understanding and consent to data processing. Neither system collects or analyzes third-party data.

Educational Intent: This system is designed as supplementary educational tools for self-improvement. They are not intended to replace human instruction or evaluate writing for high-stakes assessments.

Conclusion and Discussion

We have presented an integrated approach to personalized writing assessment in TutorBot. We believe it demonstrates that ChatGPT conversation logs serve as a valuable data source for writing assessment, with a four-stage hybrid pipeline effectively balancing classification accuracy with computational efficiency. It has approximately a 90% API cost reduction through hybrid heuristic-LLM pipeline design, it has moved beyond reactive tutoring to a system that initiates and guides learning, it has paved the way for context systems by adapting to individual student profiles and L1 interference patterns, and it has allowed for client-side processing and local tools maintaining user data sovereignty.

Limitations

Current limitations include a dependence on users having substantial ChatGPT writing history, processing time constraints from API rate limits, evaluation quality dependent on LLM capabilities, too much of a focus on the English language (limiting usability for certain parts of the world), and heuristic filtering may miss writing content using unconventional terminology. Additionally it will not re-create questions once a revaluation is done, it will only base it on new input.

Future Work

Future enhancements include:

1. **Full Database Persistence:** Complete PostgreSQL + pgvector integration for persistent learning history and adaptive spaced repetition.
2. **Expanded Tool Suite:** Implementation of the full tool suite including detect_real_task, schedule_review, and adjust_agent_complexity.
3. **Voice Integration:** Adding speech-to-text and text-to-speech for pronunciation practice and natural voice conversation.
4. **Cross-Platform Support:** Supporting additional conversation export formats (Claude, Gemini) and implementing local LLM options.

5. **Formal Evaluation:** Conducting user studies to validate learning outcomes and compare the effectiveness of pipeline versus agent approaches.

AI Usage Declaration

Claude (Anthropic) was used to assist with code development, specially when it came to web interaction. All system design decisions, architectural choices, and research directions were made by the human authors. Generated code and text were reviewed and modified as needed to ensure accuracy and alignment with project goals.

Appendix A: Code Repositories

The source code for both systems is publicly available on GitHub:

1. TutorBot (Pipeline-based Writing Assessment): https://github.com/ShahaanK/writing-improvement-ui
2. Sage (Conversational Agent Tutor): dev-ai-kar/sage

Appendix B: Heuristic

```
const writingKeywords = [
    // Core writing & editing
    'write', 'writing', 'edit', 'editing', 'proofread', 'revise',
    'revision', 'rewrite', 'rephrase', 'grammar', 'punctuation',
    'tone', 'sentence', 'paragraph', 'draft', 'feedback', 'format',

    // Academic writing
    'essay', 'paper', 'report', 'document', 'analysis', 'argument',
    'summary', 'thesis', 'dissertation', 'abstract', 'introduction',
    'conclusion', 'reflection', 'response', 'notes', 'jot notes',
    'chapter summary', 'quote integration', 'bibliography',
    'reference', 'citation', 'cite', 'peer review', 'manuscript',
    'publication', 'academic', 'formal', 'professional',

    // Creative writing
    'story', 'scene', 'narrative', 'creative', 'prompt',

    // Professional writing
    'email', 'letter', 'cover letter', 'linkedin', 'message',
    'statement', 'personal statement', 'supplementary essay',
    'application writing', 'proposal', 'presentation', 'speech',

    // Common writing task phrases
    'expand', 'shorten', 'make it human', 'sound more human'
  ];

  // Only filter EXTREMELY obvious non-writing content
  const excludeKeywords = [
    // Coding & technical
    'debug this code', 'syntax error', 'compile error', 'stack trace',
    'test coverage', 'unit test', 'function definition',
    'code', 'python', 'javascript', 'sql', 'html', 'css', 'racket',
    'algorithm', 'compute', 'compile', 'schema', 'erd', 'erd diagram',
    'normal form', '1nf', '2nf', '3nf', 'query',

    // Math / calculation
    'solve for x', 'calculate the', 'find the derivative',
    'solve this equation', 'what is the integral',
    'limit', 'derivative', 'vector', 'matrix', 'complex number',
```

```
  // Physics / science
  'velocity', 'acceleration', 'force', 'momentum', 'energy',
  'projectile', 'work', 'kinetic', 'potential',
  'molecule', 'reaction', 'glycolysis', 'atp',

  // Fitness
  'workout routine', 'sets and reps', 'bench press program',
  'workout', 'squat', 'deadlift', 'reps', 'sets', 'cardio',

  // Tests & quizzes
  'quiz', 'test', 'multiple choice', 'practice problems',
  'solve for', 'answer key',

  // Entertainment
  'recipe for', 'how to cook', 'movie recommendation',
  'game walkthrough', 'song lyrics', 'game', 'movie', 'song', 'video'
];
```

Appendix C: Prompt

```
     const prompt = `You are a writing evaluation expert. Analyze each of the
following ${batch.length} messages for grammar, punctuation, and tone quality.

For EACH message, provide:
1. Grammar Score (1-5): 1=many errors, 5=perfect grammar
2. Punctuation Score (1-5): 1=many errors, 5=perfect punctuation
3. Tone Score (1-5): 1=very inappropriate, 5=perfectly appropriate
4. Specific issues found (be concise)

Messages:
${messagesText}

Respond with ONLY a valid JSON array of ${batch.length} evaluation objects in
the SAME ORDER as the messages.
NO markdown, NO explanations, ONLY the JSON array:

[
  {
    "message_number": 1,
    "grammar_score": 3,
    "punctuation_score": 4,
    "tone_score": 4,
    "grammar_issues": ["subject-verb disagreement in sentence 2", "tense
inconsistency"],
    "punctuation_issues": ["missing comma after introductory phrase"],
    "tone_issues": ["slightly too casual for context"]
  }
]`;
```