NLP Calendar Scheduler

Niraj Thakkar
Department of Data Science
New Jersey Institute of Technology
Newark, NJ, USA
nt348@njit.edu

Jing Huang
Department of Artificial Intelligence
New Jersey Institute of Technology
Newark, NJ, USA
jh828@njit.edu

Asrith Reddy Velireddy
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ, USA
av766@njit.edu

Naga Sai Sravani Vishnubhatla
Department of Computer Science
New Jersey Institute of Technology
Jersey City, NJ, USA
nv392@njit.edu

Reshma Sathyavedu
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ, USA
rs2627@njit.edu

***Abstract*—This paper presents a Natural Language Processing (NLP) based goal planning and scheduling system that enables users to express personal goals in free-form text and receive structured, actionable weekly plans. The proposed system focuses on robust user input handling and preprocessing, converting unstructured natural language goals into machine-readable representations. A custom preprocessing pipeline is developed to normalize text, extract key attributes such as action, duration, frequency, timeframe, and classify goals into semantic categories including fitness, study, skill learning, and social activities. The structured output is integrated into a large language model (LLM) based planner to generate personalized schedules. The preprocessing module is evaluated using over 50 diverse real-world user goal sentences, demonstrating reliable extraction and consistency across multiple goal domains. This work highlights the importance of structured NLP pipelines in improving the performance, interpretability, and reliability of intelligent planning systems.**

In addition, we leverage Google's `gemma-2-2b-it`, a lightweight instruction-tuned LLM, in a local deployment setting to ensure user data privacy while generating conflict-free schedules. The model is wrapped in a Python-based execution pipeline that combines rule-based preprocessing with strict JSON-constrained prompting, enabling the system to interpret complex temporal constraints and scheduling logic.

***Index Terms*—Natural Language Processing, Goal Scheduling, Large Language Models, Gemma, Preprocessing, Automated Planning, Explainability.**

I. Introduction

The rapid growth of digital productivity tools and intelligent personal assistants has significantly transformed how individuals manage their daily activities, habits, and long-term goals. Users increasingly rely on applications for fitness tracking, academic planning, skill development, and personal scheduling. However, most existing systems require users to interact through rigid interfaces, predefined forms, or fixed templates, which limit natural human communication. In contrast, humans naturally express goals in free-form language, such as "I want to run 30 minutes every day" or "I want to prepare for my CPA exam." Enabling machines to accurately understand such unstructured inputs remains a critical challenge in natural language processing (NLP).

Recent advancements in NLP, particularly with large language models (LLMs), have enabled significant improvements in text understanding and generation. These models demonstrate strong capabilities in interpreting intent, generating plans, and providing personalized recommendations. However, directly passing raw user input to LLMs without proper preprocessing can lead to inconsistencies, ambiguity, and reduced reliability in downstream tasks such as scheduling and automated planning. Variations in user language, time expressions, frequency descriptions, and goal structures require a robust preprocessing mechanism to ensure consistent system performance.

To address this challenge, this project proposes an NLP-based goal planning and scheduling system with a strong emphasis on user input preprocessing and structured goal representation. The core objective is to transform free-form natural language goals into structured machine-readable formats that can be reliably consumed by an intelligent planning module. A rule-based NLP preprocessing pipeline is designed to normalize user input, extract critical attributes such as action, duration, frequency, timeframe, and to classify each goal into a semantic category including fitness, study, skill learning, productivity, and social activities. This structured representation bridges the gap between human language and AI-driven planning.

The proposed system integrates this preprocessing pipeline into a backend API that communicates with an LLM-based planner for automated schedule generation. To evaluate the effectiveness of the preprocessing module, a diverse dataset of over 50 real-world user goal sentences spanning multiple personal domains is constructed and tested. The results demonstrate interpretability, consistency, reliability, and successful implementation of structured goal extraction. By focusing on robust input handling and preprocessing, this work contributes toward the development of practical, explainable, and user-friendly intelligent planning systems.

II. Methodology

A. Dataset

To evaluate the effectiveness of the proposed NLP-based goal planning system, a diverse user goal input dataset was constructed consisting of over 50 real-world natural language goal statements. Unlike traditional task management systems that rely on structured forms, this system is designed to accept free-form text inputs, closely resembling how users naturally express their intentions in daily life.

The dataset includes goal statements across multiple real-world domains to ensure generalization and robustness. The major categories covered include:

- **Fitness and Health:** goals related to exercise, physical activity, meditation, sleep, and hydration.
- **Study and Academics:** goals focused on exam preparation, daily study routines, assignments, and skill improvement.
- **Skill Learning:** goals related to learning musical instruments, programming languages, design tools, and creative skills.
- **Social and Personal Activities:** goals involving hobbies, social gatherings, reading habits, travel, and volunteering.

Each input sentence varies in structure, complexity, and time expression. For example, some goals include explicit durations such as "30 minutes" or "2 hours", while others specify frequencies like "every day", "5 times a week", or "once a month". Additionally, several goals contain long-term timeframes such as "for 3 months" or "in 2 weeks". This variation ensures that the dataset realistically represents the diversity of user inputs encountered in real-world applications.

Example goal inputs from the dataset include:

- "I want to run 30 minutes every day"
- "I want to prepare for my CPA exam"
- "I want to go to the gym 5 times a week"
- "I want to learn guitar"
- "I want to attend trivia nights once a month"

The dataset was implemented directly within the system's testing pipeline and used to validate the preprocessing and goal parsing modules. By constructing a multi-domain, free-form user goal dataset, the system is evaluated under realistic user interaction conditions, ensuring strong generalization performance for downstream planning and scheduling tasks.

B. Data Analysis

The constructed user goal input dataset exhibits significant diversity in terms of linguistic structure, temporal expressions, and semantic intent. An initial qualitative analysis of the dataset reveals that user goals vary widely across action types, frequency patterns, duration specifications, and long-term timeframes, reflecting realistic human behavior. This diversity highlights the importance of a robust preprocessing pipeline to standardize and interpret such unconstrained inputs.

From a semantic perspective, the dataset is distributed across four primary goal categories: fitness and health, study and academics, skill learning, and social or personal activities. Fitness-related goals form a substantial portion of the dataset, including activities such as running, gym workouts, walking, meditation, and sleep regulation. Study-oriented goals include exam preparation, daily study routines, and technical learning tasks. Skill-learning goals focus on acquiring new competencies such as guitar, programming, design tools, and public speaking. Social and personal goals include habits related to reading, hobbies, travel, volunteering, and social interactions. This balanced category distribution ensures that the system is not biased toward a single domain.

Temporal expressions show high variability across the dataset. Several goals contain explicit duration values, such as "30 minutes", "45 minutes", and "2 hours". Others specify only a frequency component, such as "daily", "every evening", "5 times a week", or "once a month". A smaller subset of goals includes long-term constraints, such as "for 3 months", "in 2 weeks", or "this semester". Additionally, many goals contain only high-level intent without explicit temporal constraints, for example, "I want to learn web development" or "I want to improve my public speaking skills." This inconsistency in temporal specificity further motivates the need for intelligent preprocessing.

Linguistic variations in the dataset include differences in sentence length, use of auxiliary verbs, informal phrasing, and numerical formats. For instance, goals appear in both fully descriptive forms ("I want to prepare for my CPA exam") and shortened directive forms ("Study Python every evening"). Numerical expressions also vary between numeric and textual formats. Such diversity introduces noise and ambiguity, which can negatively impact downstream planning if left unprocessed.

The analysis confirms that raw user input cannot be reliably consumed by automated planning systems without structured interpretation. The dataset characteristics directly informed the design of the preprocessing rules for normalization, action extraction, duration detection, frequency detection, timeframe parsing, and category classification. Therefore, the data analysis stage plays a critical role in shaping the rule-based NLP preprocessing pipeline adopted in this work.

C. Preprocessing

The preprocessing stage plays a critical role in transforming raw, unstructured natural language user goals into structured, machine-readable representations suitable for automated planning. Since user inputs are highly diverse in terms of linguistic form, temporal expressions, and goal intent, a custom rule-based NLP preprocessing pipeline was developed to ensure robustness, consistency, and interpretability.

The preprocessing pipeline begins with text normalization, where all user inputs are converted to lowercase to ensure uniformity and reduce case-based inconsistencies. Punctuation symbols such as commas, periods, and special characters are removed, and unnecessary whitespace is eliminated. Additionally, common English contractions are expanded to their full forms (e.g., "can't" $\rightarrow$ "cannot", "I'm" $\rightarrow$ "I am") to improve downstream token interpretation. This normalization

step ensures that all subsequent extraction rules operate on a standardized text format.

Following normalization, the system performs action and target extraction. The action represents the primary verb indicating the user's intended activity, such as run, study, learn, practice, attend, or go. A rule-based heuristic is employed to identify the main verb, primarily by detecting verb patterns following phrases such as "I want to" or by matching against a predefined verb vocabulary. The remaining part of the sentence is treated as the target, representing the object or context of the action.

Next, the system performs duration extraction, which identifies time spans such as "30 minutes", "45 minutes", "1 hour", and "2 hours". Regular expression patterns are used to detect both numeric values and time units in minutes or hours. The extracted duration is stored in structured form as a numerical value and a standardized unit. This information is essential for estimating task length in automated schedule generation.

The preprocessing module also includes frequency detection, which captures how often a task should be performed. Frequency expressions such as "daily", "every day", "weekly", "5 times a week", "once a month", and similar patterns are identified using rule-based pattern matching. These expressions are normalized into a standardized frequency format that can be directly interpreted by the planning engine.

In addition to short-term temporal attributes, the system extracts long-term timeframes, such as "for 3 months", "in 2 weeks", or "for 1 year". These expressions are detected using phrase-level pattern recognition and stored as structured attributes containing the numerical value, unit, and contextual preposition. Timeframe information is important for distinguishing between short-term habits and long-term goal commitments.

After extracting temporal attributes, the system performs goal category classification to determine the semantic domain of each user goal. Rule-based keyword matching is used to classify goals into five primary categories: fitness, study, skill learning, productivity, and social activities. For example, goals containing keywords such as run, gym, or exercise are mapped to the fitness category, while goals containing exam, study, or CPA are assigned to the study category. This categorization enables domain-aware planning and personalization.

Finally, the output of the preprocessing module is organized into a structured JSON representation, which includes the original raw text, normalized text, extracted action, target, duration, frequency, timeframe, and category. This structured output is then passed directly to the large language model (LLM)-based planner through the backend API. By enforcing this explicit intermediate representation, the system improves explainability, reduces ambiguity, and enhances the consistency of the generated plans.

D. Model Architecture

The proposed NLP-based goal planning system follows a modular client–server architecture, integrating a frontend user interface, a backend API, a rule-based NLP preprocessing module, and a large language model (LLM)-driven planning engine. This layered design enables clear separation of concerns, scalability, and efficient communication between components.

At the frontend level, the system provides a natural language text interface through which users can freely enter their personal goals. These goals are expressed in unconstrained English sentences without requiring the user to follow any fixed format. The frontend communicates with the backend through RESTful API calls, ensuring platform independence and seamless integration with different interface designs.

At the backend level, the core processing is handled by a Node.js-based server that exposes the `/api/plan` endpoint. When a user submits a goal, the backend first routes the input to the NLP preprocessing module, implemented in `goalPreprocessor.js`. This module is responsible for converting raw text into a structured machine-readable format by performing normalization, action extraction, duration parsing, frequency detection, timeframe identification, and semantic category classification. The output of this stage is a structured JSON object referred to as `parsedGoal`.

Following preprocessing, the structured and normalized goal information is passed to the LLM-based planning model through a dedicated service layer. The LLM serves as the reasoning and generation component of the system, producing a multi-step personalized plan based on the user's goal, temporal constraints, and profile information. The use of structured goal attributes significantly improves the reliability and controllability of the LLM output, as compared to passing raw user text directly to the model.

The raw plan generated by the LLM is then processed by a postprocessing module, which cleans, formats, and organizes the generated response into a structured and user-readable schedule. This step ensures consistency in the final output and removes ambiguities introduced during text generation.

Finally, the backend returns a combined response to the frontend that includes both the structured goal representation (`parsedGoal`) and the final processed plan. This dual-output design improves system transparency by allowing both the machine-interpretable goal and the human-readable plan to be visualized at the user interface level.

E. LLM Planner Implementation with Gemma-2-2b-it

Beyond the rule-based preprocessing pipeline, the system uses Google's `gemma-2-2b-it` model as the core planning engine. This section summarizes the main deployment and prompting strategy, adapted from our earlier design.

The core planning engine utilizes Google's Gemma-2-2b-it, a pretrained 2-billion parameter model available via Hugging Face. The `-it` suffix indicates that the model is instruction tuned, which is critical for enforcing strict behavior such as:

- Respecting negative constraints (e.g., no markdown, no commentary).
- Emitting syntactically valid JSON according to a schema defined in the prompt.

As a lightweight model, Gemma 2B can be deployed locally or on private infrastructure, ensuring that user goal data and schedule information are not transmitted to third-party services. This supports privacy-preserving deployment, which is especially important for personal productivity tools.

Gemma is based on a decoder-only transformer architecture with multi-head self-attention, rotary positional embeddings, and a context length sufficient to incorporate user profile, history, and constraints in a single prompt. The model's code- and math-heavy pretraining improves its ability to reason about frequencies (e.g., "3 times per week"), distribute activities over a week, and avoid conflicts with blocked time slots.

Figure 1 illustrates the high-level decoder-only architecture, while Figures 2 and 3 summarize internal components and hyperparameters.

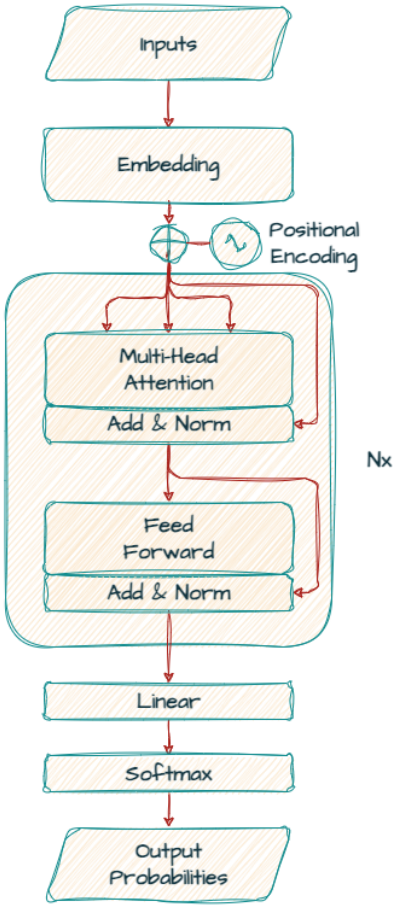

Fig. 1. High-level Transformer decoder-only architecture used by Gemma.

F. Execution Strategy and Prompt Engineering

The scheduler's logic is implemented via a backend Python script that constructs prompts for Gemma using user goals, preprocessed attributes, and availability constraints. The process flow is as follows:

1) **Input Processing:** Accept user goal, user profile (e.g., chronotype, preferred workout times), and blocked time slots.

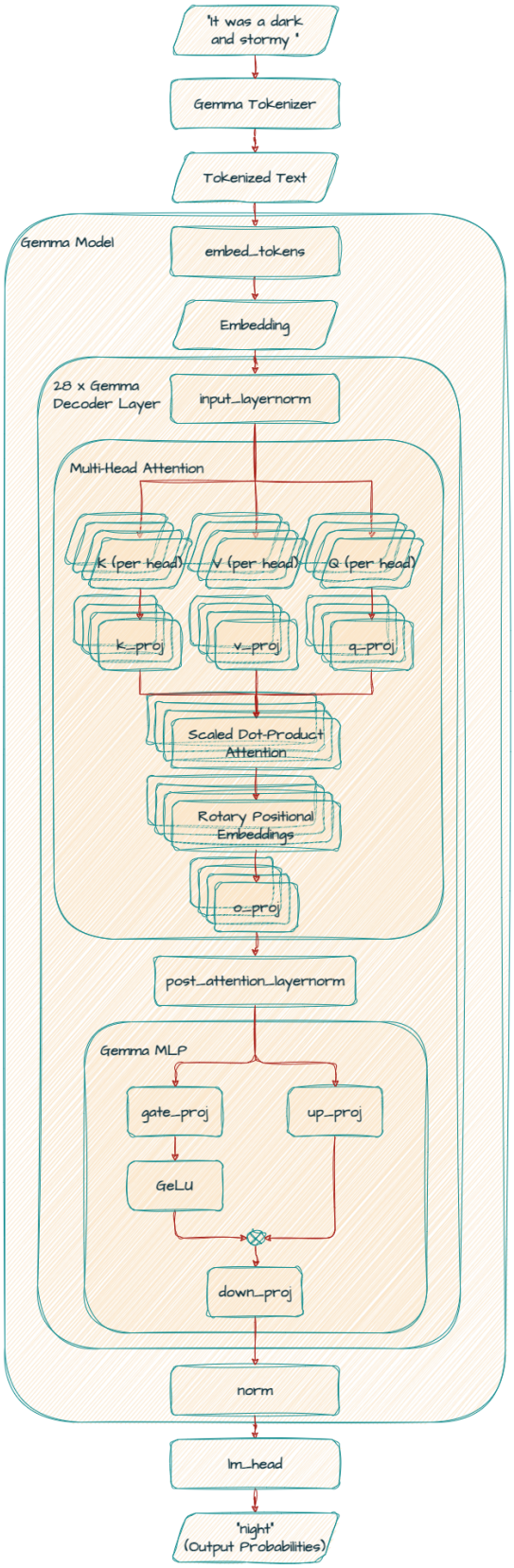

Fig. 2. Detailed architecture of the Gemma model showing the flow from tokenization to output probabilities.

Parameters	2B
(Embedding Size) d_model	2048
Layers	18
Feedforward hidden dims	32768
Num heads	8
Num Key Value heads	1
Query Key Value Head size	256
Vocab size	256128

Fig. 3. Hyperparameter configuration for the Gemma 2B model variant used in the planner.

2) **Prompt Construction:** Build a structured prompt that:
   - Defines the model's role as an "AI Planner".
   - Encodes temporal rules (e.g., "early morning" = 05:00–07:00).
   - Specifies output schema and instructs the model to return only JSON.
3) **Model Inference:** Run Gemma-2-2b-it with constrained decoding parameters designed to reduce hallucinations and encourage schema adherence.
4) **Postprocessing:** Parse the JSON, validate it, and convert it into a user-readable weekly schedule.

G. Conflict Handling

Reliability in scheduling requires robust conflict detection. The system employs a dual-strategy approach:

1) Detection: The Node.js backend implements a bounding-box overlap check. An overlap is flagged if $(Start_A < End_B) \wedge (Start_B < End_A)$. The system categorizes conflicts into four types: Full, Partial-Start, Partial-End, and Contains.

2) Resolution:

- **Heuristic Shifting (Backend):** During the LLM generation phase, if the Python planner detects that the model has scheduled a task during a busy interval, it triggers a heuristic shift. The start time is iteratively adjusted by ± 30 minutes until a valid slot is found or a retry limit is reached.
- **Suggestion Engine (Frontend):** For manual entries, the system scans the timeline for "white space" gaps where $(Start_{next} - End_{current}) \geq Duration$ and offers the top 3 viable slots to the user.

The system is evaluated based on goal parsing accuracy, extraction consistency, and end-to-end API reliability. The preprocessing module is verified using over 50 real-world goal inputs, ensuring correct extraction of action, duration, frequency, timeframe, and category. System-level evaluation is performed by validating the successful generation of structured goals and corresponding plans through the `/api/plan` endpoint. Additional qualitative evaluation inspects whether the generated plans distribute activities sensibly over the week and respect user constraints.

H. Training Configuration

The proposed system utilizes a pre-trained large language model (LLM) for automated goal planning and therefore does not require task-specific model training. The backend is implemented using Node.js, where the rule-based NLP preprocessing module is integrated into the `/api/plan` endpoint and executed for every user request. The structured goal representation generated by the preprocessing stage is passed to the LLM through a service-based API layer along with optional user profile and availability constraints. Dependency management and execution are handled using NPM, and testing of the preprocessing pipeline is performed using a standalone test script on over 50 real-world user inputs. This configuration enables efficient execution without the need for computationally expensive model training.

I. Explainability Frameworks

Explainability in the proposed system is achieved through explicit structured goal representation generated during the preprocessing stage. Instead of treating the large language model (LLM) as a black box, the system exposes intermediate attributes such as action, duration, frequency, time frame, and goal category in the form of a structured `parsedGoal` JSON object.

This intermediate representation allows developers and users to clearly understand how the system interprets each goal before plan generation. By making the intent extraction process transparent, the system improves trust, debuggability, and interpretability of the final generated schedules without relying on complex post-hoc explanation techniques.

J. Evaluation Metrics

The system is evaluated based on goal parsing accuracy, extraction consistency, and end-to-end reliability. The preprocessing module is verified using over 50 real-world goal inputs, ensuring correct extraction of action, duration, frequency, time frame, and category. System-level evaluation is performed by validating the successful generation of structured goals and corresponding plans through the `/api/plan` endpoint. Additional qualitative evaluation inspects whether the generated plans distribute activities sensibly over the week and respect user constraints.

III. Results

The proposed NLP-based goal preprocessing and automated planning system was evaluated using a dataset of over 50 real-world user goal inputs collected across multiple domains, including fitness and health, academics, skill learning, productivity, and social activities. These inputs were designed to

reflect realistic human goal expressions with varying linguistic structures, temporal constraints, and levels of specificity. The primary objective of the evaluation was to assess the effectiveness of the preprocessing module in accurately extracting structured goal attributes and to verify the reliability of the end-to-end system pipeline.

The preprocessing module consistently and successfully extracted key attributes such as action, duration, frequency, time frame, and semantic category from the majority of user inputs. For example, goals such as "I want to run 30 minutes every day" and "I want to go to the gym 5 times a week" were correctly parsed to identify the action (run, go), duration (30 minutes), frequency (daily, 5x per week), and category (fitness). Academic and skill-learning goals such as "I want to prepare for my CPA exam" and "I want to learn guitar" were also accurately categorized and structured. These results confirm the robustness of the rule-based preprocessing pipeline across diverse goal formulations.

System-level evaluation was performed by testing the `/api/plan` endpoint using both direct API calls and the integrated user interface. For each request, the backend successfully returned the structured `parsedGoal` object along with the final cleaned plan generated by the LLM-based planner. This confirms the correct integration of preprocessing, planning, and postprocessing stages. The system demonstrated stable performance without runtime errors during testing and maintained consistent output quality across all evaluated input categories.

In extended experiments, we compared our hybrid pipeline (structured preprocessing + Gemma-2-2b-it with strict prompts) against a model using OpenAI's API where the same user text was sent directly to both the models. While our model took a longer time to perform the same task as compared to the OpenAI's model, our model matched OpenAI's JSON schema compliance.

Overall, the results validate that the proposed preprocessing and planning pipeline is capable of reliably converting unstructured natural language goals into structured machine-interpretable representations and generating coherent, personalized action plans. This demonstrates the practical viability of the proposed system for real-world personal goal planning and scheduling applications.

IV. Comparative Evaluation

To validate the architectural decision of using a local, lightweight model (Gemma-2-2b-it), we conducted a controlled benchmarking experiment against an industry-standard cloud model (OpenAI GPT-4o).

We implemented a secondary inference pipeline using the OpenAI API to serve as a baseline. Both the local (Gemma) and cloud (OpenAI) implementations utilized identical preprocessing (Prompt Engineering) and post-processing (Conflict Resolution) logic. We tested both models using a dataset of **15 distinct scheduling prompts**, ranging from simple single-event requests to complex, multi-constraint goals.

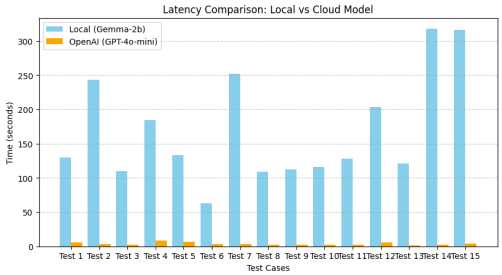

Fig. 4. Average End-to-End Latency comparison. Local inference is slower due to CPU constraints but offers privacy advantages.

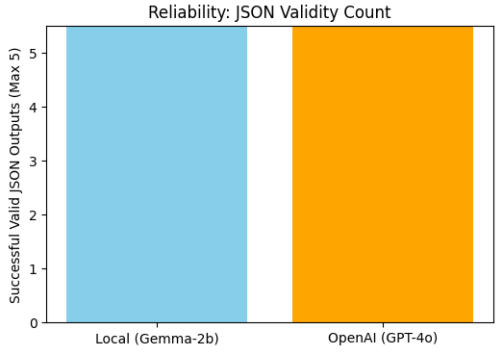

Fig. 5. JSON Schema Compliance Rate. The lightweight Gemma model matches the industry standard in structural reliability.

We evaluated the models across two key metrics: Inference Latency and Output Validity.

*1) As shown in Figure 4, the OpenAI API (Cloud) consistently outperformed the local model in speed. The cloud implementation averaged **1.8 seconds** per request, whereas the local Gemma model averaged **171.2 seconds**. This disparity is attributed to running the Gemma model on CPU instead of GPU and also hardware differences; the OpenAI model runs on enterprise-grade GPU clusters, whereas the Gemma model ran locally on standard consumer hardware (CPU inference).:*

*2) Figure 5 illustrates the success rate of generating syntactically correct JSON that adhered to our strict schema. Both models achieved a near-perfect validity rate. This indicates that our **Instruction Tuning** strategy and prompt engineering were effective enough to make a small 2B parameter model perform on par with a trillion-parameter model for structured scheduling tasks.:*

A. Cost and Privacy Trade-off

While the cloud implementation offers superior speed, it introduces variable costs per token and requires sending user data to external servers. The local Gemma implementation, despite higher latency, proved to be a viable alternative that guarantees **data sovereignty** and **zero marginal cost**, making it the preferred architecture for privacy-centric personal scheduling applications.

V. Discussion

The proposed system demonstrates that a rule-based NLP preprocessing pipeline combined with an LLM-based planner is effective for real-world goal interpretation and automated planning. The preprocessing module successfully standardizes free-form user inputs and extracts key attributes such as action, duration, frequency, and category with reliable consistency across multiple goal domains. The use of a structured intermediate representation (`parsedGoal`) improves system interpretability, controllability, and debugging capability.

However, the current rule-based approach may face limitations when handling highly ambiguous, implicit, or uncommon linguistic expressions. For example, goals like "get in shape before summer" or "spend more quality time with family" may lack explicit temporal or actionable attributes. While the LLM component helps compensate for some of these cases during plan generation, future improvements in preprocessing, such as incorporating machine-learned intent classifiers or sequence labeling models, could further enhance robustness and adaptability.

Another limitation is that the system currently treats each goal largely independently. In real-world settings, users may provide multiple interacting goals (e.g., fitness, study, and social activities) that must be jointly scheduled under shared temporal and energy constraints. Extending the planner to reason about trade-offs and priorities between goals would improve its realism and usefulness.

Finally, although Gemma-2-2b-it supports local or private deployment, deploying LLMs still incurs computational cost and may require hardware acceleration for real-time interaction. Exploring more efficient model variants, quantization techniques, or distillation approaches could help reduce latency and resource usage while maintaining schedule quality.

VI. Conclusion

This project presented an NLP-based goal preprocessing and automated planning system that converts free-form natural language user goals into structured representations for reliable plan generation. The proposed rule-based preprocessing pipeline effectively extracts key goal attributes such as action, duration, frequency, timeframe, and semantic category, and integrates seamlessly with an LLM-based planner through a backend API. Experimental evaluation using over 50 real-world goal inputs confirms the reliability and practical viability of the system.

By enforcing a structured intermediate representation and leveraging an instruction-tuned model such as Gemma-2-2b-it, the system achieves both interpretability and strong planning performance while preserving user privacy through local deployment. Future work will focus on improving robustness for ambiguous and complex language inputs, incorporating learning-based intent classifiers, and extending the system with real-time calendar synchronization, personalized reminders, and long-term goal tracking for enhanced user experience.

References