SMATIKA : STIKI Informatika Jurnal
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 79/E/KPT/2023
E-ISSN: 2580-6939 || P-ISSN: 2087-0256

Large Language Models for JSON-Based Function Call Planning from Indonesian Natural Language: A Restaurant Search Chatbot Case Study

Mohammad Mauludin[1*], Joan Santoso[2], Hartarto Junaedi[3]

[1,2,3]*Institut Sains dan Teknologi Terpadu Surabaya, Faculty of Science and Technology, Departement of Information Technology, Jalan Ngagel Jaya Tengah 73 - 77, Baratajaya, Kec. Gubeng, Surabaya, Jawa Timur 60284, Indonesia*

Keyword

Large Language Models; Function Call Planning; JSON Generation; Indonesian Natural Language; Conversational Agents

Corresponding Author:
mauludin.artuji@gmail.com

Abstract

Large Language Models are increasingly adopted as planning components that translate natural language into structured representations for tool invocation, enabling executable interaction with backend systems through JSON-based function calling. However, empirical studies focusing on Indonesian natural language remain limited. This paper presents a restaurant search chatbot case study that investigates JSON-based function call planning from Indonesian user queries, with emphasis on the upstream planning task rather than conversational response generation. A synthetic dataset of 33,470 Indonesian restaurant search queries paired with ground truth JSON plans was constructed based on a predefined tool set and database schema. Supervised fine-tuning with parameter-efficient adaptation was applied to a pretrained language model. The fine-tuned Mistral 7B model was evaluated using multiple metrics measuring JSON structural validity, tool sequence correctness, and parameter accuracy at different granularities. The results demonstrate strong performance, achieving a JSON structure validity rate of 0.97, tool sequence exact match accuracy of 0.92, column-level accuracy of 0.97, and value-level accuracy of 0.94, while session-level evaluation highlights remaining challenges in composing all parameters correctly within a single planning instance. Overall, this study shows that with carefully designed datasets and strict supervision, Large Language Models can reliably perform structured JSON-based function call planning from Indonesian natural language, supporting their applicability to structured application domains where execution correctness is critical.

1. Introduction

Large Language Models have recently emerged as a central component in modern natural language processing systems, demonstrating strong capabilities in understanding, reasoning, and generating human like text[1], [2].Beyond conventional language generation tasks, Large Language Models are increasingly used as reasoning engines that can plan actions and interact with external systems through structured outputs[3]. One important mechanism enabling this interaction is JSON based function calling, where a model translates natural language input into a structured JSON representation that specifies function names and parameters to be executed by backend services[4], [5]. This paradigm allows conversational systems to bridge free-form human language with deterministic programmatic operations, moving beyond traditional rule-based approaches and systems that heavily rely on third-party services for intent processing and execution[6]. In practical applications, JSON based function call planning has become a key building block for tool augmented chatbots, intelligent assistants, and autonomous agents[7]. By generating structured plans instead of direct textual answers, a language model can delegate concrete tasks such as database querying, filtering, ranking, or information retrieval to external tools while maintaining flexibility in user interaction. This approach improves system modularity, interpretability, and reliability, especially in domains that require precise constraint handling and consistent execution logic.

Restaurant search is a representative domain where natural language interaction and structured execution must coexist. Beyond convenience, this domain is particularly suitable because it reflects a real-world conversational recommender system scenario in which user preferences are complex, multi-dimensional, and often evolve through interaction. More importantly, restaurant search encapsulates core challenges common to many structured application domains, including constraint composition, parameter grounding to a fixed schema, and the need for precise execution over backend data sources. Users typically express their needs in informal and flexible language, including preferences related to location, budget, facilities, ambience, and personal taste. Translating such requests into structured queries therefore requires not only understanding the semantic content of the utterance but also planning which functions to invoke and how to parameterize them correctly in order to support accurate recommendation and retrieval processes. Traditional rule based or form driven systems often struggle to capture this flexibility, making conversational systems powered by Large Language Models an attractive alternative[8]. However, most existing studies and implementations of language model based function calling focus primarily on English[4] [9]. This creates a significant research gap for languages such as Bahasa Indonesia, which differ in syntax, pragmatics, and discourse structure. Indonesian natural language frequently relies on implicit subjects, flexible word order, and colloquial expressions, all of which pose challenges for consistent and accurate extraction of structured parameters[10]. Applying English centric configurations directly to Indonesian inputs often results in malformed JSON outputs or incorrect function planning, limiting the practical usability of such systems.

This study addresses this gap by presenting a case study on Large Language Models for JSON based function call planning from Indonesian natural language in the context of a restaurant search chatbot. The main objective of this work is to analyze how effectively a language model can transform Indonesian user queries into well structured JSON plans that represent function calls required by a backend restaurant search system. Rather than evaluating end to end recommendation quality, this study focuses on the upstream planning problem, namely the translation of user intent into machine executable function calls. The proposed chatbot architecture positions the Large Language Model as a planner that interprets user input and generates JSON based function call plans. These plans specify which backend functions should be executed, such as filtering restaurants by city or price range, retrieving detailed restaurant information, or ranking candidates according to user preferences. This design is inspired by recent agent oriented frameworks that separate reasoning and planning from tool execution[3],[7]. By decoupling natural language understanding and deterministic execution, the system achieves clearer behavioral control while retaining conversational flexibility. A central aspect of this study is supervised fine tuning as a means to adapt a pretrained language model to Indonesian language usage and strict JSON output requirements. While general purpose language models show promising zero shot performance, they often fail to consistently produce syntactically valid and semantically accurate JSON structures without domain specific adaptation[11]. To address this issue, supervised fine tuning is conducted using a synthetic dataset of Indonesian restaurant search interactions, where each user utterance is paired with a target JSON function call plan. Parameter efficient fine tuning techniques such as Low Rank Adaptation are employed to enable effective adaptation with limited computational resources[12].

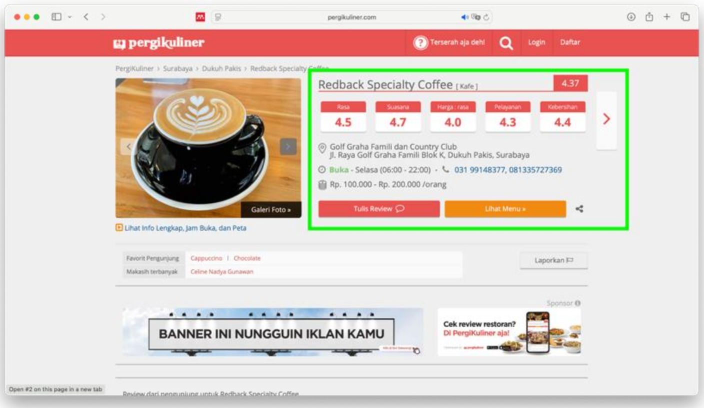
This study addresses the following research questions: (1) to what extent can a supervised fine-tuned Large Language Model accurately translate Indonesian natural language queries into structurally valid and executable JSON-based function call plans in a restaurant search chatbot; (2) how well can the model understand user intents and generate function parameters that correctly reflect the users' intended constraints and preferences; and (3) how frequently does the model produce planning errors in tool selection, sequencing, and parameterization during function call planning. The contributions of this paper are threefold. First, it analyzes the challenges of JSON-based function call planning from Indonesian natural language, with emphasis on linguistic characteristics that affect structured output generation. Second, it presents a restaurant search chatbot design as a concrete case study for formulating function call planning using Large Language Models in a non-English setting. Third, it documents a supervised fine-tuning strategy that adapts a pretrained language model to produce well-structured JSON plans that conform to predefined schemas and planning constraints. By limiting the scope of discussion to model analysis and fine-tuning, this paper provides a focused reference on adapting Large Language Models for Indonesian natural language to JSON planning tasks. Through this case study, we demonstrate that with appropriate dataset design and supervised fine-tuning, Large Language Models can reliably perform JSON-based function call planning from Indonesian natural language, enabling robust and scalable conversational systems for real-world restaurant search applications.

2. Research Method

This study adopts a design-oriented experimental research methodology aimed at systematically analyzing and adapting Large Language Models for JSON-based function call planning from Indonesian natural language. The methodology positions the Large Language Model as a planning component within a tool-augmented conversational system and focuses on evaluating its ability to translate natural language inputs into structured and executable JSON plans. Rather than proposing a new model architecture, this work emphasizes controlled dataset construction, supervised fine-tuning, and fine-grained evaluation to study planning behavior and error patterns. The methodological design follows the practical workflow of building a real-world conversational recommender system, while deliberately limiting the scope to model preparation, planning supervision, and structured output evaluation. Accordingly, the overall process consists of data collection, system design, dataset creation, supervised fine-tuning, and quantitative evaluation.

2.1 Data Collection

The data collection process in this study is designed to support restaurant search within the scope of JSON based function call planning from Indonesian natural language. The domain is explicitly limited to restaurant search, where users issue natural language queries to find restaurants that satisfy specific constraints such as location, price range, facilities, payment options, and qualitative preferences. This focus aligns with the objective of analyzing how Large Language Models translate Indonesian user queries into structured JSON plans for backend function execution in a restaurant search chatbot. Restaurant data was collected from PergiKuliner, a widely used Indonesian culinary review platform that provides detailed restaurant information and user generated reviews written in Bahasa Indonesia. PergiKuliner was selected because it offers semi structured restaurant metadata that closely reflects real world restaurant search behavior in Indonesia, ensuring consistency between the language used by users and the attributes available for backend filtering and retrieval. A visual representation of the PergiKuliner website is presented in Figure 1.


pergikuliner.com
pergikuliner
Terserah aja deh!
Login
Daftar
PergiKuliner › Surabaya › Dukuh Pakis › Redback Specialty Coffee
Galeri Foto »
Lihat Info Lengkap, Jam Buka, dan Peta
Laporkan
Sponsor
BANNER INI NUNGGUIN IKLAN KAMU
Cek review restoran? Di PergiKuliner aja!
Open #2 on this page in a new tab
Review dari pengunjung untuk Redback Specialty Coffee

Redback Specialty Coffee [Kafe] 4.37

Rasa	Suasana	Harga : rasa	Pelayanan	Kebersihan
4.5	4.7	4.0	4.3	4.4

Golf Graha Famili dan Country Club
Jl. Raya Golf Graha Famili Blok K, Dukuh Pakis, Surabaya
Buka - Selasa (06:00 - 22:00) · 031 99148377, 081335727369
Rp. 100.000 - Rp. 200.000 /orang
Tulis Review
Lihat Menu »

Favorit Pengunjung	Cappuccino | Chocolate
Makasih terbanyak	Celine Nadya Gunawan

Figure 1. PergiKuliner Webpage

Data acquisition was performed through automated web scraping of publicly accessible restaurant detail pages on the PergiKuliner website, specifically targeting the information displayed within the green highlighted sections shown in Figure 1. The scraping process focused on individual restaurant pages to ensure consistent and complete extraction of all attributes required for restaurant search operations. Only restaurant-level information was collected, and no personal or identifiable information related to reviewers was stored or processed. Duplicate entries were removed based on restaurant name and address similarity, and restaurants with missing critical search attributes such as location or price range were excluded to ensure data consistency and reliability.

To support controlled experimentation and detailed analysis of JSON-based function call planning, the restaurant database was intentionally limited to 100 restaurant entries. This design choice enables manual cross-checking and human verification of generated JSON plans against ground truth during dataset construction and evaluation, which would be impractical at larger database scales. A bounded restaurant set facilitates systematic inspection of tool selection, parameter correctness, and execution order, ensuring the reliability of supervision signals used in fine-tuning. In addition, limiting the database size reduces computational overhead during training and inference, while allowing the large synthetic query dataset to emphasize linguistic diversity and constraint composition rather than database scale. Despite this limitation, the selected restaurants were curated to cover diverse locations, price ranges, facilities, payment methods, and rating profiles, ensuring that the database remains representative of common restaurant search scenarios in Indonesia.

From each restaurant entry, a fixed set of structured attributes was extracted to form the core database schema used by the restaurant search backend. The collected fields include the city, district, full address, restaurant name, minimum and maximum price range, phone number, overall rating, and aspect specific ratings covering food flavor, atmosphere, price suitability, service quality, and cleanliness. In addition, categorical attributes describing available facilities and accepted payment methods were collected. Facilities include options such as 100% Halal, 24 Jam, Area outdoor, Area parkir, Bisa reservasi, Pesan antar, Ruang VIP, and Wifi, while payment methods include Dana, Debet, GoPay, Kartu, Master, Ovo, QRIS, Transfer, Tunai, and Visa. These attributes correspond directly to constraints commonly expressed in Indonesian restaurant search queries and are used by backend functions such as filtering, ranking, and sorting.

In addition to structured attributes, user review texts associated with each restaurant were collected as unstructured data to support semantic search operations. Reviews often contain implicit information relevant to restaurant search, such as ambience, dining experience, or suitability for certain occasions, which cannot be fully captured through predefined structured fields. All reviews for a given restaurant were aggregated and transformed into vector embeddings, which are stored separately and accessed only by semantic retrieval tools during planning execution. Before storage, the collected data underwent normalization and cleaning to ensure consistency and deterministic backend behavior. Textual fields such as city and district names were standardized to reduce spelling variations. Price values were converted into numeric representations, and rating values were normalized into a consistent decimal scale. Categorical attributes such as facilities and payment methods were mapped to predefined controlled vocabularies to eliminate ambiguity during filtering. After normalization, the resulting dataset forms a structured and queryable restaurant search database that can be reliably accessed by backend functions and serves as the foundation for dataset creation and supervised fine tuning.

2.2 System Design

The architecture of the proposed system is designed to explicitly study JSON-based function call planning from Indonesian natural language. In this architecture, the chatbot operates as a planning agent, where the primary responsibility of the Large Language Model is to generate a structured plan that determines which backend tools should be invoked and in what order[13]. The system does not aim to generate conversational responses, instead the final output of the system is the result produced by the executed tools, as shown in Figure 2.

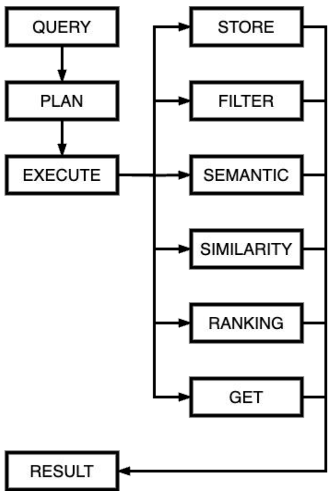


Figure 2. Architecture of the planning-oriented restaurant search chatbot.

As illustrated in Figure 2, the interaction starts with a user query expressed in Indonesian natural language. The query is passed to a planning agent, where a Large Language Model (LLM) performs a planning step before any backend execution. In this phase, the LLM interprets user intent and constraints and generates a JSON-based function call plan that specifies the selected tools and their parameters. The architecture consists of three stages: Query, Plan, and Execute. The Query stage receives the raw input, the Plan stage translates the input into a structured JSON plan, and the Execute stage performs the specified tool calls strictly according to the generated plan, without further involvement of the LLM. The system provides a predefined set of deterministic backend tools, including Store, Filter, Semantic, Similarity, Ranking, and Get, each exposing a fixed JSON schema. These tools support operations such as storing user preferences, filtering candidates by structured constraints, semantic refinement, similarity search, ranking, and retrieving detailed restaurant information.

Depending on query complexity, the planner may produce single-step or multi-step plans. Correctness therefore depends on both tool selection and execution order, which are explicitly encoded in the JSON plan. The final output is the result of the executed tools, with no additional natural language generation, ensuring that evaluation focuses solely on structured planning accuracy rather than response fluency.

2.3 Dataset

The dataset used in this study is designed to support JSON based function call planning for a restaurant search chatbot operating on Indonesian natural language input. Unlike conventional conversational datasets that emphasize natural language response generation, this dataset focuses exclusively on the planning task, where the objective of the model is to translate user queries into structured JSON plans that specify backend tool invocations and their execution order. Due to the absence of publicly available datasets that align Indonesian restaurant search queries with explicit function call plans, a synthetic dataset was constructed. The dataset creation process follows the system architecture described earlier, where each sample represents a single planning instance grounded in the predefined tool set and restaurant database schema[9],[14]. A teacher model was employed to generate realistic Indonesian user queries together with their corresponding JSON based tool plans. This approach ensures that all generated plans are executable and strictly conform to the JSON schemas required by the backend tools. Rather than relying on fixed or manually crafted query templates, the teacher model generates natural language queries conditioned on structured intent and parameter specifications. As a result, lexical and syntactic variation naturally emerges from the generative process, producing semantically equivalent queries with different surface forms while preserving the same underlying planning intent and JSON structure. Although no explicit rule-based paraphrasing or post-generation lexical augmentation was applied, this generation strategy reduces the risk of template memorization and encourages the model to learn robust mappings from Indonesian natural language to structured function call plans.

Each dataset instance consists of two main components: a natural language query in Bahasa Indonesia and a structured JSON plan produced by the assistant. The JSON plan encodes one or more tool invocations, where each invocation specifies the tool name and the parameters required for execution. For queries that require multiple backend operations, tool calls are included within a single JSON structure and ordered sequentially to represent the intended execution flow. The dataset deliberately excludes natural language responses derived from tool outputs, as the training objective is restricted to structured planning rather than conversational response generation. To ensure consistency during supervised fine tuning, all dataset samples are serialized using a dialogue style format that explicitly separates the user input and the planning output. An example of the dataset structure is shown in Figure 3.

```
<user>
Eh, ada nggak sih restoran yang rating kebersihannya 4 ke atas?
</user>
<assistant>
{
  "function_calls": [
    {
      "name": "filter",
      "arguments": [
        {
          "operand": "null",
          "condition": [
            {
              "column": "rating_clean",
              "operator": ">=",
              "value": 4.0
            }
          ]
        }
      ]
    },
```

Figure 3. Dataset Structure

Figure 3 representation provides a clear supervision signal, enabling the model to learn a direct mapping from Indonesian natural language input to valid JSON based function call plans. The dataset is organized around several planning scenarios that reflect different aspects of function call planning complexity in restaurant search.

1) Tool Understanding
Designed to train the model to recognize and correctly invoke the available backend tools, including Get, Filter, Similarity, Semantic, Store, and Ranking, based on user intent. In this scenario, the model learns to select the appropriate tool and generate the required parameters in JSON format. For example, a query requesting restaurants in a specific city should trigger the Filter tool with the corresponding city constraint.

2) Structure Understanding

Focuses on enabling the model to interpret and utilize structured attributes defined in the restaurant database schema, such as city, district, address, price range, ratings, facilities, and payment methods. The model is expected to accurately extract these attributes from natural language queries and encode them into function call parameters. For instance, when a user asks for restaurants in Jakarta with a price below a certain threshold, the model must correctly identify both the city and price constraints and represent them in the JSON plan.

3) Comparative
Addresses queries that involve multiple constraints combined using logical operators such as "and" and "or." This scenario trains the model to correctly interpret compound conditions and translate them into structured filtering logic. For example, when a user asks for restaurants that are budget friendly and offer free Wi-Fi, the model must apply both conditions simultaneously within the generated function call plan.

2.4 Supervised Fine-Tuning

The supervised fine tuning process was conducted to adapt the pretrained Large Language Model for JSON based function call planning in the context of Indonesian language restaurant search. The primary objective of this stage was not to improve natural language response quality, but to align the model's outputs with structured planning requirements, enabling it to reliably translate Indonesian user queries into valid and executable JSON plans that specify backend tool invocations and their execution order.

1) Data Preprocessing
In the preprocessing stage, the planning dataset was organized into structured interaction pairs between the user and the planning agent[15]. Each interaction was formatted using a chat style template that explicitly encodes message roles through special tokens such as <|user|> and <|assistant|>. This chat-based formatting has been shown to be effective for aligning pretrained language models with task-specific behaviors during supervised fine tuning[16],[17]. All user queries are provided in Bahasa Indonesia, while the assistant outputs consist exclusively of JSON based function call plans. Text inputs were tokenized and truncated to fit within the maximum context length supported by the model[18]. A padding token was added to ensure consistent sequence lengths across training batches. During training, only the assistant tokens corresponding to the JSON plan are used as labels, while user input tokens are masked out and excluded from loss computation. This masking strategy ensures that the model is trained solely to generate structured outputs and has been widely adopted in instruction tuning and structured generation settings[19].

2) Low-Rank Adaptation (LoRA)
To enable efficient adaptation without updating the full set of model parameters, parameter efficient fine tuning was applied using Low Rank Adaptation[12]. LoRA modules were injected into selected attention and projection layers, including q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj. These layers are critical for learning intent recognition, tool selection, and parameter extraction in structured planning tasks.
The LoRA configuration uses a rank of 16, a scaling factor of 32, and a dropout rate of 0.05. The LoRA rank parameter was set to $r = 16$, which determines the dimensionality of the low-rank adaptation matrices. This value was selected as a compromise between adaptation capacity and computational efficiency. A rank value that is too small may limit the model's ability to effectively adapt to new task-specific patterns, while a larger rank increases the number of additional parameters and memory requirements. By setting $r = 16$, the model achieves sufficient adaptation flexibility for structured JSON-based function call planning without incurring a significant increase in computational or memory overhead. This configuration allows the model to adapt to Indonesian language patterns and strict JSON generation constraints while maintaining computational efficiency. By freezing the original model weights and training only the low rank adapters, the fine tuning process remains lightweight and stable.

3) Training Hyperparameters
Fine tuning was conducted for three epochs with a learning rate of 1e−5. The choice of three epochs was motivated by the characteristics of parameter-efficient fine tuning with LoRA, where task adaptation is typically achieved within a small number of passes over the data. Preliminary training observations showed

that the training and validation loss stabilized within the first two epochs, indicating that additional epochs provided limited performance gains while increasing the risk of overfitting to patterns[20]. A per device batch size of 2 was used, combined with gradient accumulation over 8 steps to effectively utilize GPU memory. The AdamW optimizer was employed with a weight decay of 0.01 to promote generalization. Mixed precision training using FP16 was applied to accelerate training and reduce memory consumption. To further safeguard against overfitting, an early stopping callback was enabled during fine tuning. Training was automatically terminated if the validation metric failed to improve for two consecutive evaluation intervals. As a result, the three-epoch configuration functions as an upper bound rather than a fixed training duration, ensuring that the final model is selected based on optimal validation performance rather than the final epoch.

4) Data Collator
A custom dynamic data collator was implemented to support efficient training on variable length planning sequences. The collator applies dynamic padding based on the longest sequence within each batch, minimizing redundant padding while maintaining uniform tensor dimensions. It produces input_ids, attention_mask, and labels tensors for each batch. The attention mask differentiates valid tokens from padding tokens, ensuring that computation focuses only on meaningful input. Padding positions within the label tensor are replaced with −100 so that they are ignored during loss calculation. This setup guarantees that gradient updates are driven exclusively by the correctness of the generated JSON based function call plans.

2.5 Evaluation

The evaluation aims to assess the ability of Large Language Models to perform JSON based function call planning from Indonesian natural language in a restaurant search setting. Two pretrained models were evaluated in this study: Mistral 7B Base Model and Komodo 7B Base Model. Both models were fine tuned using the same dataset, architecture, and supervised fine tuning procedure to ensure a fair comparison. Evaluation was conducted using a fixed set of held out test queries that were not seen during training.

Each evaluation run consists of 100 independent test queries written in Bahasa Indonesia. The test set size was intentionally limited to 100 sessions to maintain a controlled and computationally feasible evaluation process. As a result, evaluating a substantially larger number of sessions would significantly increase inference time and computational cost, and may lead to timeout issues during testing. Despite this limitation, the selected test queries were designed to represent diverse planning patterns, including single-step and multi-step tool sequences as well as categorical and numeric constraints, ensuring adequate coverage of the planning task.

For each query, the model generates a JSON based function call plan as its output. The generated plan is then compared against a manually defined ground truth plan that specifies the correct tool selection and execution order. The evaluation does not involve executing the backend tools or assessing the quality of retrieved restaurant results. This exclusion is intentional in order to isolate the planning capability of the model from downstream components such as database completeness, retrieval mechanisms, or ranking strategies. By decoupling planning from execution, the evaluation ensures that errors can be attributed specifically to failures in intent understanding, tool selection, ordering, or parameter generation, rather than external system factors.

To achieve this, the evaluation relies on a set of quantitative metrics that assess both the structural validity of the generated JSON and the correctness of the planned tool sequence. The following metrics are used to evaluate model performance.

1) JSON Structure Validity Rate
JSON Structure Validity Rate measures the model’s ability to generate outputs that conform to the required JSON format for function call planning, focusing solely on structural correctness independent of execution or semantic accuracy. Evaluation is performed on 200 test sessions, consisting of two sets of 100 model outputs representing different restaurant search planning scenarios. Each session corresponds to a single planning

output generated from an Indonesian natural language query. An output is considered valid if it is a well-formed JSON object and contains a function_calls field structured as a list of tool invocations. Outputs with malformed JSON or missing required fields are classified as invalid. Let N denote the total number of evaluation sessions and N_valid denote the number of sessions that produce structurally valid JSON outputs. The JSON Structure Validity Rate is defined as:

$$JSON\ Structure\ Validity\ Rate = \frac{N_{valid}}{N} \tag{1}$$

2) Tool Sequence Exact Match Accuracy

Tool Sequence Exact Match Accuracy is used to evaluate the model's ability to generate the correct sequence of tool invocations for a given user query. This metric focuses on the planning aspect of JSON based function call generation, specifically assessing whether the model can correctly determine which tools should be called and in what order. A session is considered correct if and only if the predicted tool sequence exactly matches the ground truth tool sequence, both in terms of tool names and execution order. Any mismatch in tool selection, sequence length, or ordering is counted as an incorrect planning outcome. Tool Sequence Exact Match Accuracy is defined as:

$$Tool\ Sequence\ Exact\ Match\ Accuracy = \frac{N_{correct}}{N} \tag{2}$$

3) Session-level Parameter Exact Match Accuracy

Session-level Parameter Exact Match Accuracy is used to evaluate the model's ability to correctly generate filter parameters required for structured restaurant search. This metric operates at the session level, providing a binary correctness signal for each evaluation instance. A score of 1 is assigned only if all filter parameters exactly match the ground truth, while a score of 0 is assigned if any single parameter is incorrect, missing, duplicated, or if parsing fails at any stage. Session-level Parameter Exact Match Accuracy is defined as:

$$Session - level\ ParameterExact\ Match\ Accuracy = \frac{N_{correct}}{N} \tag{3}$$

4) Parameter-level Exact Match Count

Parameter-level Exact Match Count is used to evaluate the model's ability to correctly generate individual filter parameters at a finer granularity than session-level metrics. Unlike Session-level Parameter Exact Match Accuracy, which assigns a binary score to an entire session, this metric counts how many ground truth filter parameters are exactly matched by the predicted parameters across all evaluation sessions. If a model output fails to produce valid JSON, lacks a Filter tool invocation, or cannot be parsed correctly, the session contributes zero correct parameter matches. Parameter-level Exact Match Count accuracy is defined as:

$$Parameter - level\ Exact\ Match\ Accuracy = \frac{\sum_i C_i}{\sum_i G_i} \tag{4}$$

5) Operator-level Exact Match

Operator-level Exact Match is used to evaluate the model's ability to correctly predict comparison operators associated with filter parameters in restaurant search queries. This metric focuses specifically on operator correctness, independent of parameter values or tool execution order, and assesses whether the model selects the appropriate logical operator required for accurate backend filtering. An operator prediction is considered correct only if it exactly matches the ground truth operator for the same parameter. Operator-level Exact Match Accuracy is defined as:

$$Operator - level\ Exact\ Match\ Accuracy = \frac{\sum_i C_i}{\sum_i G_i} \tag{5}$$

6) Value-level Exact Match Accuracy

Value-level Exact Match Accuracy is used to evaluate the model's ability to correctly predict parameter values associated with filter conditions in JSON based function call planning. For each session, the predicted filter values generated by the model are compared against the corresponding ground truth values for the same parameters. A value prediction is considered correct only if the parameter name matches and the value exactly matches the ground truth after normalization. Value-level Exact Match Accuracy is defined as:

$$Value - level\ Exact\ Match\ Accuracy = \frac{\sum_i C_i}{\sum_i G_i} \tag{6}$$

7) Column-level Exact Match Accuracy

Column-level Exact Match Accuracy is used to evaluate the model's ability to correctly identify which database columns (parameter names) should be used in filter conditions during JSON based function call planning. A column prediction is considered correct only if the predicted column name exactly matches the corresponding ground truth parameter name. Column-level Exact Match Accuracy is defined as:

$$\text{Column} - \text{level Exact Match Accuracy} = \frac{\sum_i C_i}{\sum_i G_i} \tag{7}$$

3. Result and Discussions

This section presents the experimental results from supervised fine tuning and evaluation of models for JSON-based function call planning from Indonesian natural language. The analysis focuses on the models' ability to learn structured planning behavior, maintain JSON validity, and generalize to unseen restaurant search queries. Rather than evaluating recommendation quality, the discussion emphasizes planning-level behavior, including training stability and adherence to predefined tool schemas.

1) Model Training

This subsection analyzes the supervised fine-tuning results for JSON-based function call planning from Indonesian natural language. The analysis focuses on how effectively pretrained models adapt to structured planning, particularly in enforcing strict JSON constraints and learning multi-step tool invocation logic. Two models, Mistral 7B Base Model and Komodo 7B Base Model, are fine-tuned using identical datasets, LoRA configurations, and training hyperparameters to ensure a fair comparison. Training and evaluation loss curves are used to examine learning dynamics, convergence behavior, and generalization to unseen planning instances. Figures 4 and 5 present the training and evaluation loss comparisons across training steps for both models.

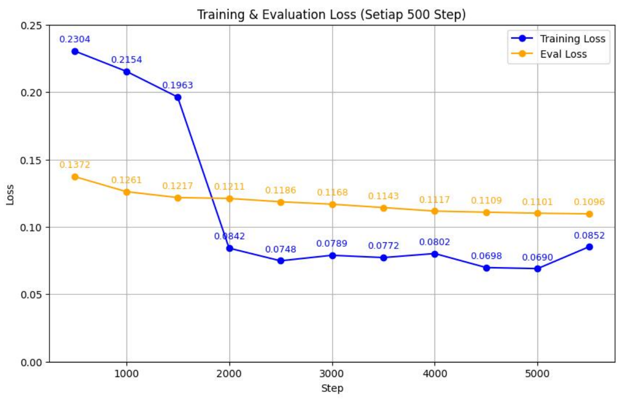


Figure 4. Training and Evaluation Loss Curves for Mistral 7B Base Model

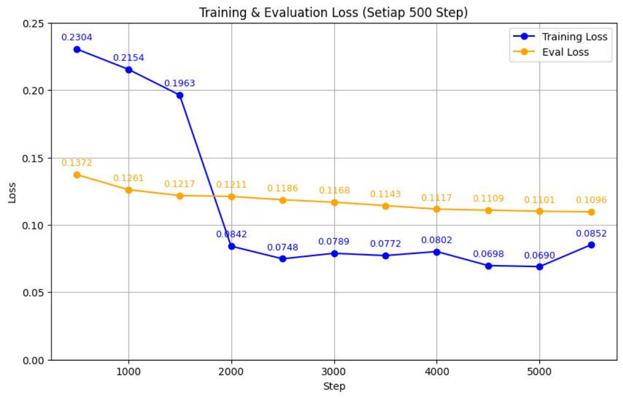


Figure 5. Training and Evaluation Loss Curves for Comodo 7B Base Model

As shown in Figure 4 and Figure 5, both models exhibit a consistent decrease in training loss during the initial stages of fine tuning, indicating that they successfully learn the mapping between Indonesian user queries and structured JSON based function call plans. This behavior suggests that the dataset and supervision signal are effective in guiding the models toward the intended planning behavior, including tool selection and parameter extraction. However, differences between the models become apparent as training progresses. Mistral 7B demonstrates more stable convergence, with evaluation loss decreasing steadily and remaining relatively consistent throughout the later stages of training. This stability indicates that the model generalizes well to unseen queries and maintains adherence to strict JSON formatting and planning constraints. Such robustness is particularly important in function call planning tasks, where minor deviations can result in invalid or non executable outputs. In contrast, Komodo 7B shows signs of overfitting after extended training. While its training loss continues to decrease, the evaluation loss begins to increase after a certain number of training steps. This divergence suggests that the model becomes increasingly specialized to the training data and less effective at generalizing to unseen planning scenarios. In the context of JSON based function call planning, this behavior can lead to higher rates of malformed JSON or incorrect tool sequencing when handling new user queries.

Overall, the training loss analysis indicates that supervised fine tuning with LoRA is effective for adapting Large Language Models to Indonesian JSON based planning tasks. At the same time, the comparison highlights that model stability during training plays a critical role in determining downstream planning reliability. The observed results suggest that Mistral 7B is better suited for structured function call planning, as it demonstrates more consistent convergence and stronger generalization under strict output constraints.

2) Evaluation Results

To provide a consolidated overview of model performance, the evaluation results for Mistral 7B Fine Tuned and Komodo 7B Fine Tuned across all defined metrics are summarized in Table 1. The table reports performance at multiple levels, including JSON structural validity, tool sequence planning, and parameter correctness at session, parameter, operator, value, and column levels. This comprehensive set of metrics enables systematic comparison of the two models under increasingly strict evaluation criteria.

Tabel 1. Evaluation Results

Metrics	Mistral-7B	Komodo-7B
JSON Structure Validity Rate	0.97	0.94
Tool Sequence Exact Match Accuracy	0.92	0.84
Session-level Parameter Exact Match Accuracy	0.41	0.08
Parameter-level Exact Match Count	0.76	0.44
Operator-level Exact Match	0.78	0.52
Value-level Exact Match Accuracy	0.94	0.81

Column-level Exact Match Accuracy	0.97	0.91

Based on the results presented in Table 1, the following discussion analyzes each evaluation metric in detail. The analysis begins with JSON Structure Validity Rate as the foundational requirement for executable planning, and then progressively examines tool sequencing accuracy and parameter correctness at finer levels of granularity. To complement the quantitative findings, representative qualitative examples of model outputs are provided in the appendices.

Appendix A contains example outputs generated by the Mistral 7B base (pretrained, non fine-tuned) model. These examples illustrate typical failure patterns observed prior to supervised adaptation, including malformed JSON structures, incorrect or missing tool invocations, improper execution order, and incomplete or inconsistent parameter specification. The appendix highlights how the base model struggles to consistently adhere to predefined tool schemas and strict structural constraints required for executable planning.

Appendix B contains corresponding examples generated by the same model after supervised fine tuning using the proposed dataset and LoRA configuration. These outputs demonstrate improved structural validity, correct tool sequencing, and more accurate parameter extraction aligned with the predefined JSON schema. By comparing Appendix A and Appendix B, the qualitative evidence reinforces the quantitative improvements reported in Table 1 and illustrates how supervised fine tuning enhances planning reliability in JSON based function call scenarios.

1) JSON Structure Validity Rate.
Mistral 7B achieves a JSON Structure Validity Rate of 0.97, outperforming Komodo 7B, which attains a rate of 0.94. This result indicates that both models are generally capable of producing syntactically valid and schema compliant JSON outputs after supervised fine tuning. However, the higher validity rate of Mistral 7B reflects stronger robustness in adhering to strict structural constraints. In practical JSON based function call planning, this robustness is critical, as structurally invalid outputs cannot be executed by backend systems and are treated as complete planning failures.

2) Tool Sequence Exact Match Accuracy.
In terms of tool planning, Mistral 7B achieves a Tool Sequence Exact Match Accuracy of 0.92, significantly higher than Komodo 7B, which reaches 0.84. This difference demonstrates that Mistral 7B is more reliable in generating the correct sequence of tool invocations, particularly for multi step queries that require combining structured filtering with semantic refinement. Correct tool ordering is essential for executable planning, and the observed gap indicates that Komodo 7B more frequently produces planning sequences that deviate from the intended execution logic.

3) Session-level Parameter Exact Match Accuracy.
The largest performance gap is observed in Session-level Parameter Exact Match Accuracy, where Mistral 7B achieves 0.41, while Komodo 7B attains only 0.08. This strict metric requires all filter parameters within a session to match the ground truth exactly. Under this formulation, a single incorrect parameter causes the entire session to be counted as a failure. The very low score of Komodo 7B indicates that in most sessions at least one parameter is incorrectly generated, resulting in complete planning failure under strict evaluation. In contrast, although Mistral 7B achieves substantially higher performance, a score of 0.41 still shows that composing fully correct multi-parameter plans remains challenging even for the stronger model.

4) Parameter-level Exact Match Count.
At the parameter level, Mistral 7B achieves 0.76 accuracy, compared to 0.44 for Komodo 7B. The gap between parameter-level and session-level accuracy indicates that multiple parameter errors often occur within the same session. For Komodo 7B, error analysis shows that many parameter mismatches are caused by normalization and formatting issues rather than complete intent misinterpretation. For example, the model sometimes generates city names without proper spacing in multi-word locations, leading to exact-match

failure despite near-correct semantic understanding. In other cases, numeric inputs such as phone numbers beginning with "081" are incorrectly transformed into values such as 81, resulting in mismatched value fields. Because the evaluation applies strict exact matching on parameter name, operator, and normalized value, such formatting or numeric transformation errors are counted as incorrect parameters. These parameter-level mismatches subsequently contribute to the very low Session-level Parameter Exact Match Accuracy, since any single incorrect parameter causes the entire session to be marked as incorrect.

5) Operator-level Exact Match.
For operator prediction, Mistral 7B attains an Operator-level Exact Match Accuracy of 0.78, whereas Komodo 7B reaches 0.52. A common error observed in Komodo 7B involves misinterpreting strict lower-bound constraints. For example, when the user specifies a condition equivalent to "kurang dari" or "di bawah," which should be translated into a strict < operator, the model occasionally generates <=. In restaurant search scenarios, this distinction may appear minor; however, semantically it changes the filtering boundary and can include unintended results.

6) Value-level Exact Match Accuracy.
At the value level, both models perform relatively well, with Mistral 7B achieving 0.94 and Komodo 7B achieving 0.81. The narrower gap at this level suggests that extracting numeric thresholds and categorical values from Indonesian natural language is less challenging than selecting operators or composing parameters. Nevertheless, Mistral 7B still demonstrates superior consistency, indicating more reliable value normalization and interpretation across diverse query formulations.

7) Column-level Exact Match Accuracy.
Finally, Column-level Exact Match Accuracy shows that both models possess strong schema awareness, with Mistral 7B achieving 0.97 and Komodo 7B achieving 0.91. This result suggests that both models are generally capable of identifying relevant database attributes from user queries. However, the higher accuracy of Mistral 7B reflects better alignment between natural language expressions and database schema elements, reducing the likelihood of downstream parameter and operator errors.

4. Conclusions and Future Works

This study demonstrates that Large Language Models can be effectively adapted for JSON-based function call planning from Indonesian natural language using supervised fine-tuning with strict structural supervision. Through a restaurant search chatbot case study, the results show that Mistral 7B consistently outperforms Komodo 7B across all evaluation metrics, achieving 97% JSON Structure Validity, 92% Tool Sequence Exact Match, 41% Session-level Parameter Exact Match, 76% Parameter-level Accuracy, 78% Operator-level Accuracy, 94% Value-level Accuracy, and 97% Column-level Accuracy. These results indicate that while both models can generate syntactically valid JSON, Mistral 7B exhibits substantially stronger robustness under strict planning constraints, particularly in multi-step tool sequencing and joint parameter composition, which are critical for executable backend planning. The analysis further reveals that planning performance degrades as evaluation strictness increases, emphasizing the importance of stable training dynamics and strong schema alignment, as most failures arise from operator and parameter composition errors rather than value extraction alone.

Despite these promising results, several limitations must be acknowledged. First, the training dataset is synthetically generated using a teacher model, which may not fully capture the variability, ambiguity, and noise present in real-world user interactions. Second, the study is restricted to a single application domain, namely restaurant search, with a fixed backend schema; therefore, the findings may not directly generalize to other domains with different schema structures or execution logic. Third, the experimental comparison is limited to two 7B-scale models (Mistral 7B and Komodo 7B), and does not include larger-scale models, alternative architectures, or zero-shot baselines, which may influence comparative conclusions regarding fine-tuning effectiveness. Consequently, while the results demonstrate strong planning capability within the defined scope, broader generalization requires further empirical validation.

Future research can extend this approach beyond the restaurant domain to other structured application areas such as product search, travel booking, healthcare information retrieval, and customer support workflows, enabling the study of schema generalization and cross-domain transfer. Investigating how a model fine-tuned on one tool schema adapts to new domains and execution logic will provide valuable insight into the scalability and general applicability of Large Language Model-based planning agents in real-world, execution-critical systems.

5. References

[1] C. Qu *et al.*, "Tool Learning with Large Language Models: A Survey," *Front. Comput. Sci.*, vol. 19, no. 8, pp. 1–33, Nov. 2024, doi: 10.1007/s11704-024-40678-2.

[2] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, Oct. 2019, Accessed: Dec. 22, 2025. [Online]. Available: https://arxiv.org/abs/1910.10683v4

[3] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," Jun. 2021, Accessed: Dec. 22, 2025. [Online]. Available: https://arxiv.org/abs/2106.09685v2

[4] J. Lian, Y. Lei, X. Huang, J. Yao, W. Xu, and X. Xie, "RecAI: Leveraging Large Language Models for Next-Generation Recommender Systems," *WWW 2024 Companion - Companion Proceedings of the ACM Web Conference*, vol. 1, pp. 1031–1034, Mar. 2024, doi: 10.1145/3589335.3651242.

[5] J. Wei *et al.*, "Finetuned Language Models Are Zero-Shot Learners," *ICLR 2022 - 10th International Conference on Learning Representations*, Sep. 2021, Accessed: Dec. 22, 2025. [Online]. Available: https://arxiv.org/abs/2109.01652v5

[6] S. AlFahryan and S. Suryayusra, "Pengembangan Website Chatbot Untuk Kampus Bina Darma," *SMATIKA JURNAL*, vol. 13, no. 02, pp. 304–317, Dec. 2023, doi: 10.32664/SMATIKA.V13I02.930.

[7] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, pp. 815–824, Aug. 2016, doi: 10.1145/2939672.2939746.

[8] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback".

[9] L. Wang *et al.*, "A Survey on Large Language Model based Autonomous Agents," *Front. Comput. Sci.*, vol. 18, no. 6, pp. 1–42, Mar. 2025, doi: 10.1007/s11704-024-40231-1.

[10] OpenAI *et al.*, "GPT-4 Technical Report," Mar. 2023, Accessed: Dec. 21, 2025. [Online]. Available: https://arxiv.org/abs/2303.08774v6

[11] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," *11th International Conference on Learning Representations, ICLR 2023*, Oct. 2022, Accessed: Dec. 21, 2025. [Online]. Available: https://arxiv.org/abs/2210.03629v3

[12] T. Auliarachman and A. Purwarianti, "Coreference Resolution System for Indonesian Text with Mention Pair Method and Singleton Exclusion using Convolutional Neural Network," *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, Sep. 2020, doi: 10.1109/ICAICTA.2019.8904261.

[13] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *Adv. Neural Inf. Process. Syst.*, vol. 2020-December, May 2020, Accessed: Dec. 21, 2025. [Online]. Available: https://arxiv.org/abs/2005.14165v4

[14] T. Schick *et al.*, "Toolformer: Language Models Can Teach Themselves to Use Tools," *Adv. Neural Inf. Process. Syst.*, vol. 36, Feb. 2023, Accessed: Dec. 21, 2025. [Online]. Available: https://arxiv.org/abs/2302.04761v1

[15] A. Abdullah, Jumadi, and D. Firdaus, "Implementasi Algoritma Bidirectional Encoder Representations From Transformer Pada Speech To Text Untuk Notulensi Rapat," *SMATIKA JURNAL*, vol. 15, no. 02, pp. 423–431, Dec. 2025, doi: 10.32664/SMATIKA.V15I02.1725.

[16] Y. Lu *et al.*, "Learning to Generate Structured Output with Schema Reinforcement Learning," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 4905–4918, 2025, doi: 10.18653/V1/2025.ACL-LONG.243.

[17] Y. Qin *et al.*, "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs," *12th International Conference on Learning Representations, ICLR 2024*, Jul. 2023, Accessed: Dec. 22, 2025. [Online]. Available: https://arxiv.org/abs/2307.16789v2

[18] M. A. Albany, I. Taufik, and I. Budiman, "Implementasi Algoritma BERT untuk Question and Answer System Terkait Hadist dalam Bentuk Virtual Youtuber," *SMATIKA JURNAL*, vol. 15, no. 02, pp. 408–422, Dec. 2025, doi: 10.32664/SMATIKA.V15I02.1704.

[19] Y. Wang *et al.*, "Self-Instruct: Aligning Language Models with Self-Generated Instructions," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 13484–13508, Dec. 2022, doi: 10.18653/v1/2023.acl-long.754.

[20] H. Suhendar *et al.*, "Analisis Sentimen Hasil Transkripsi Audio Berbahasa Indonesia Menggunakan T5 (Text-to-Text Transfer Transformer)," *SMATIKA JURNAL*, vol. 15, no. 01, pp. 115–125, Jun. 2025, doi: 10.32664/SMATIKA.V15I01.1521.