Extracting Parallelism from Large Language Model Queries

Steven Kolawole*
Carnegie Mellon University
skolawol@andrew.cmu.edu

Keshav Santhanam*
Stanford University
keshav2@stanford.edu

Virginia Smith
Carnegie Mellon University
smithv@cmu.edu

Pratiksha Thaker*
Carnegie Mellon University
pthaker@andrew.cmu.edu

Abstract

Optimization engines for LLM query serving typically focus on workloads with known structure, treating the query itself as a black box. In this work, we investigate extracting parallelization opportunities from individual queries that have decomposable subtasks. Using the `LMSYS-chat-1M` dataset, we identify three query categories that are amenable to decomposition into parallel LLM calls, and curate a dataset of these queries as a benchmark for this type of within-query parallelization. We develop a prototype system to parallelize these queries and report initial performance results, showing that parallelization can result in a speedup of $5\times$ over serial execution with comparable or even improved generation quality.

1 Introduction

Interactive interfaces for large language models (LLMs) are increasingly becoming a part of end users' daily workflows, replacing search engines and contributing generated text. In order to support these use cases, interactive LLM interfaces must serve diverse, unstructured queries at interactive latencies. Unfortunately, LLM serving systems to date have largely been focused on identifying optimization opportunities in highly structured batch workloads in which the query structure is known in advance and an optimized pipeline can be tailored to the query of interest.

In this paper, we observe that users of interactive LLM interfaces submit queries that have optimization opportunities *within* the initial query itself, embedded in the language. Executing these queries naïvely as a single call to the LLM may miss important optimization opportunities present in the query. As a simple example, asking a language model to generate ten short stories will intuitively take longer than making ten language model calls for a single short story in parallel (e.g., see Figure 1).

Developers have built ad-hoc systems to take advantage of parallelism opportunities within natural language [e.g., Shenoy and Derhacobian, 2024], but these systems are tailored to particular query types and require users to provide structured queries (essentially asking users to provide the structure needed for extracting opportunites for parallelism).

Within the `LMSYS-chat-1M` dataset [Zheng et al., 2023a], an open-source dataset of real user queries, we found hundreds of examples of parallelizable queries written in raw natural language. Using the capabilities of the language model itself and in-context examples of query structure, we can extract the parallel structure from these queries rather than requiring users to identify the structure in their queries and format them uniformly.

*Equal contribution.

NeurIPS 2024 Workshop on Adaptive Foundation Models

```
Generate 10 variations of detailed descriptions of a room, describing
     the type of room, the style, and the included furniture. The description
     is based on the following list: ["bed", "table", "nightstand", "lamp", "mirror"]
```

(a) An example of a repeated-execution query. The 10 generations can be executed in parallel.

```
How acceptable are  the following English sentences on a scale of 1 to 10?

1. The book is brown.
2. The book are brown.
3. The books on the table is brown.
4. The books on the table are brown.
5. The books that are on the table is brown.
6. The books that are on the table are brown.
```

(b) An example of a classification query. The task (rating sentences) can be parallelized across the queries (each sentence).

Figure 1: Examples of parallelizable natural language queries.

In this paper, we show that there are many common user queries that support parallelism and that they can be served in a unified interface rather than a collection of ad-hoc systems. We contribute a dataset of these examples curated from the `LMSYS-chat-1M` dataset, focusing on three main types of queries that can be unified into a single data-parallel execution interface: repeated generation, reading comprehension, and keyword extraction. We build a prototype system to parallelize and serve these queries and show that executing these queries in a data-parallel, rather than serial, way can improve performance to around $3\times$. When accounting for the number of tokens generated, we observe even greater gains, with improvements of up to $5.7\times$. Furthermore, we evaluate the quality of the outputs and find that parallelizing queries can lead to higher-quality responses within the same generation time, underscoring both the efficiency and the effectiveness of our approach. We hope this work will be a useful starting point for future work that aims to study parallelization in LLM queries.

2 Related Work

Chain-of-Thought models. The work most closely related to ours is Skeleton-of-Thought [Ning et al., 2024]. This framework is likely to be useful for repeated execution queries but less useful for queries like keyword extraction. Additionally, Ning et al. [2024] evaluate on existing datasets that are not specifically testing parallelization performance. A number of works focus on structured generation to improve generation quality, including the classic Chain-of-Thought [Wei et al., 2022] and extensions such as Topologies of Reasoning [Besta et al., 2024]. Branch-Solve-Merge [Saha et al., 2023] also finds parallel task decompositions within a query, but with the goal of improving query output for tasks like quality evaluation by assigning subtasks to different LLM agents. Parsel [Zelikman et al., 2023] provides modular decompositions specifically for programming and algorithmic reasoning tasks.

Programming models for LLM queries. Existing work on programming models for LLM queries generally focuses on simplifying prompt engineering for complex tasks. DSPy Khattab et al. [2022] provides a framework to express retrieval-augmented generation tasks and automates the task of prompting models effectively. Other frameworks provide similar functionality [Dong et al., 2024, Chase, 2022, Okuda and Amarasinghe, 2024]. Kim et al. [2023] and Singh et al. [2024] develop systems to optimize the execution of external API calls (such as to search or calculator tools) made during a query execution.

Optimization engines for LLM inference. Finally, given a structured batch LLM workload, one can design systems to efficiently serve queries (without inspecting the semantics of the query itself). ALTO [Santhanam et al., 2024] and Teola [Tan et al., 2024] use a known query graph to optimize execution. Hydragen [Juravsky et al., 2024] caches attention computation across shared query prefixes. Finally, systems like vLLM [Zheng et al., 2023b] and SGLang [Kwon et al., 2023] optimize KV cache use for general LLM programs.

3 Dataset

We release a dataset consisting of parallelizable questions from the `LMSYS-chat-1M` dataset [Zheng et al., 2023a]. We manually curated queries matching one of three parallelizable query categories:

- Repeated generation: Given a fixed prompt, generate n diverse responses to the prompt.
- Reading comprehension: Given a short passage or context, answer a series of questions based on the passage.
- Keyword extraction: Given a passage and a set of keyword names, extract elements of the passage corresponding to the keywords.

This is not an exhaustive list of possibly parallelizable, real-world queries. In Section 6, we also mention other potentially parallelizable query categories, but in our current work we focus on curating a high-quality benchmark focused on these three categories.

While the queries we curated from the dataset are representative of the tasks in each category, many are not clean or standardized in format. In order to facilitate benchmarking and pinpoint performance issues, we also generated synthetic datasets for the reading comprehension and keyword extraction datasets (we found that repeated generation queries were both plentiful in the data and also generally clean and easy to parse, so the benefit of synthetic data was minimal). We used GPT-4o with two one-shot examples each to generate a diverse set of 100 synthetic queries in each category and release these alongside the real data. In Table 1 we list the final size of the query set for each setting.

Query type	Real queries	Synthetic queries
Repeated generation	101	n/a
Reading comprehension	87	100
Keyword extraction	101	100

Table 1: Comparison of real and synthetic queries

Lastly, for our initial performance baseline, we cleaned and standardized the real queries into JSON format using a combination of manual inspection and automated translation using `GPT-4o`. In particular, we prompted `GPT-4o` to 1) clean up the initial raw queries from the `LMSYS-chat-1M` dataset by polishing the formatting and filling in anonymized entities, and 2) decompose the queries into the structured data parallel format, which we provide in Appendix A. The "`original`" entry of the schema represents the raw, unmodified version of the query taken from `LMSYS-chat-1m`, while the "`serial`" entry represents the cleaned version of the query. The "`template`", "`context`", "`data`", and "`n`" entries define the data parallel execution format. We manually created five in-context examples demonstrating the conversion of an original query into this schema for each of the three tasks.

We envision this benchmark as a series of progressively more difficult parallelization tasks – from the simplest task of parallelizing a computation already in structured format to the most complex task of deciding whether to parallelize, and then cleaning and parallelizing, a stream of unstructured queries. We present comprehensive evaluation results for the parallel execution task in §5.

4 Execution Baseline

We develop a simple performance baseline to parallelize queries in our dataset. Our observation is that many parallelizable LM queries can be represented in a common, structured schema, which we represent in our system as a JSON object 4. The schema consists of a context or template that is common across the parallel queries (such as a passage for reading comprehension queries), and a list of data to parallelize over (such as independent queries to ask about the passage). We represent repeated generation queries separately in the schema with a field for the number of repetitions.

In order to ensure some diversity of output for repeated generation queries, we implement a simple heuristic, asking for each generation to start with a different letter of the alphabet.

From a raw query string, we can prompt an LLM with few-shot examples to generate the query schema, and then execute data-parallel LM calls.

5 Evaluation

5.1 Implementation

We implemented our parallelization backend in C++, as it offers more fine-grained control over threading and concurrency; and this afforded us more efficient parallel processing, which was more

challenging to implement with Python. Through the OpenAI API, we use the `GPT-4o` model for parsing schema and `GPT-4-1106-preview` for LLM calls.

As the number of parallel calls grows significantly, we encountered challenges related to API rate limits. To address this, we implement an exponential backoff strategy that involves automatically retrying failed requests with progressively longer wait times between attempts.

5.2 Performance

We evaluated the latency improvements achieved through parallelization across the three primary tasks in both real-world and synthetic datasets. For all tasks, we measured the execution latency after the query has been converted into JSON format (a less challenging task compared to the complete, end-to-end task including conversion to the schema format) as we found that clean conversion was challenging for keyword extraction and reading comprehension. However, for "repeated generation" queries, where the conversion was relatively clean for our query set, we also include the end-to-end latency including the schema conversion.

Table 2 presents the performance metrics, including the average execution times for both the standard serial formats and the data-parallel versions, the speed-up from parallelization, and the "normalized" speedup accounting for the number of tokens per query (parallel execution typically generated longer responses).

Generally, we notice notable improvements in the execution time when parallelized. It is apparent that these tasks benefit from parallelization, as parallelization consistently outperforms serial executions across all tasks and datasets. The E2E evaluation for repeated generation, which includes the total overhead of converting raw prompts into parallelizable format, shows that the parallelized version still yields a significant speedup, with a $1.7\times$ improvement compared to the serial counterpart.

Task	Dataset Source	Avg Parallel Duration (s)	Avg Serial Duration (s)	Normalized Speedup	Speedup
Keyword extraction	`LMSYS`	2.38	3.23	**2.54**×	**1.36**×
	synthetic	1.81	3.54	**1.89**×	**1.95**×
Reading comprehension	`LMSYS`	3.49	10.27	**5.72**×	**2.94**×
	synthetic	2.96	7.48	**3.22**×	**2.52**×
Repeated generation	`LMSYS`	3.79	9.51	**4.39**×	**2.50**×
	synthetic	—	—	—	—
	`LMSYS` (E2E)	4.88	9.51	**3.41**×	**1.70**×

Table 2: Performance metrics for real (LMSYS) and synthetic data not including time for schema extraction, as well as total e2e time (including schema extraction) for repeated generation. Improvements over serial execution times are consistent across all tasks.

Scaling parallelization: To investigate the scalability of our parallelization approach, we performed an in-depth analysis using a subset of the repeated generation task. We systematically varied the value of n from 1 to 50, where n represents the number of outputs specified in a given original prompt. Figure 2 shows that we achieve the expected linear speedup from parallelization, even in spite of rate limiting (and the exponential backoff mechanism) beginning to take effect when $n > 26$.

5.3 Quality

To compare the quality of parallel and serial outputs, we use an LLM (`GPT-4o`) to judge the two versions of the generations according to their accuracy, grammar, and specificity, as well as an overall preference. In particular, our evaluation prompt included the following questions which the model had to answer by selecting either the serial generation, the parallel generation (concatenation of the individual parallel outputs), or a tie:

1. **Accuracy**: Which response more accurately follows the instructions given in the prompt?
2. **Grammar**: Which response is more grammatically accurate?

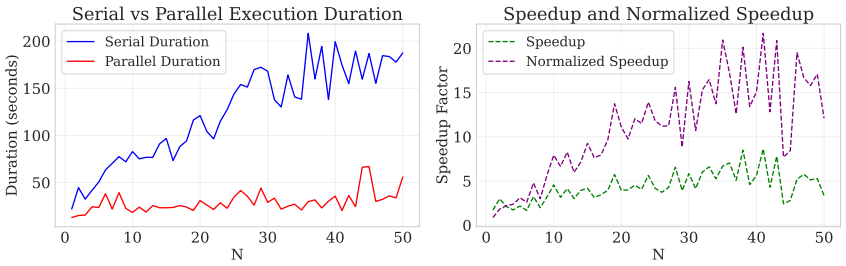
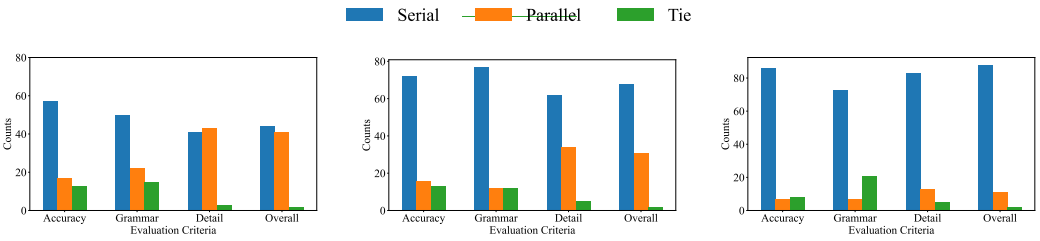


Figure 2: (a) Comparison of execution durations between serial and parallel approaches. The serial time grows linearly with n while the parallel time stays roughly constant. (b) Performance improvements achieved through parallel execution. The speedup factor, calculated as the ratio of serial to parallel execution time. Serial outputs tend to shrink and become less verbose as n increases, while parallel outputs continue to be longer and often higher quality, making the normalized speedup grow with n.

3. **Detail**: Which response provides more detail and specificity?
4. **Preference**: Which response do you personally prefer overall, considering all factors?

Figure 3 presents the results. Across the three tasks, the language model consistently judged the serial outputs to be more accurate and grammatically correct than the parallel outputs; however, its overall preferences varied significantly across tasks. For the reading comprehension task, the model showed no clear preference between serial and parallel outputs. In contrast, it exhibited a moderate preference for serial outputs in the keyword extraction task and a strong preference for serial outputs in the repeated generation task. This is because the serial outputs more closely followed the output format requested in the prompt, especially in cases where the prompt specified a particular output schema like JSON. Furthermore, the language model found the independent parallel responses for the repeated generation task to be too redundant as a result of the parallel outputs not having shared context during generation. This was not an issue for the reading comprehension task as each question can be answered independently; this also means that parallel generations enable more targeted focus on each question as evidenced by the parallel outputs scoring higher in terms of detail.



(a) Reading comprehension (b) Keyword extraction (c) Repeated generation

Figure 3: A qualitative comparison of serial generations versus parallel generations using a language model (`GPT-4o`) as a judge. We show the number of times the language model selected the serial generation or parallel generation for each evaluation question.

6 Future Work

These results suggest that a unified, data-parallel interface for LLM query execution has the potential to significantly enhance both the speed and quality of user interactions, making it a scalable solution for interactive LLM interfaces in real-world applications. However, the qualitative analysis demonstrates that a naïve parallelization approach may not be robust for queries which require specific output formats or expect independent content across parallel executions. For these queries a subsequent language model step may be necessary to assemble the parallel generations into the specified output format or filter redundant generations. We defer such exploration to future work.

References

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, et al. Topologies of reasoning: Demystifying chains, trees, and graphs of thoughts. *arXiv preprint arXiv:2401.14295*, 2024.

Harrison Chase. LangChain, October 2022. URL `https://github.com/langchain-ai/langchain`.

Honghua Dong, Qidong Su, Yubo Gao, Zhaoyu Li, Yangjun Ruan, Gennady Pekhimenko, Chris J Maddison, and Xujie Si. Appl: A prompt programming language for harmonious integration of programs and large language model prompts. *arXiv preprint arXiv:2406.13161*, 2024.

Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-throughput llm inference with shared prefixes. *arXiv preprint arXiv:2402.05099*, 2024.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. *arXiv preprint arXiv:2312.04511*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting llms for efficient parallel generation. In *The Twelfth International Conference on Learning Representations*, 2024.

Katsumi Okuda and Saman Amarasinghe. Askit: Unified programming interface for programming with large language models. In *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 41–54. IEEE, 2024.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.

Keshav Santhanam, Deepti Raghavan, Muhammad Shahir Rahman, Thejas Venkatesh, Neha Kunjal, Pratiksha Thaker, Philip Levis, and Matei Zaharia. Alto: An efficient network orchestrator for compound ai systems. In *Proceedings of the 4th Workshop on Machine Learning and Systems*, pages 117–125, 2024.

Varun Shenoy and Alex Derhacobian. Super json mode: A framework for accelerated structured output generation. `https://github.com/varunshenoy/super-json-mode`, 2024.

Simranjit Singh, Andreas Karatzas, Michael Fore, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. An llm-tool compiler for fused parallel function calling. *arXiv preprint arXiv:2405.17438*, 2024.

Xin Tan, Yimin Jiang, Yitao Yang, and Hong Xu. Teola: Towards end-to-end optimization of llm-based applications. *arXiv preprint arXiv:2407.00326*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. Parsel: Algorithmic reasoning with language models by composing decompositions. *Advances in Neural Information Processing Systems*, 36:31466–31523, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023a.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*, 2023b.

A Query Schema

Figure 4 presents the schema used to encapsulate data parallel queries.

```
{
   # Original prompt string without parallelization transform
   "serial": str = None,

   # A template for data parallel generation which may include some context
   "template": str = None,

   # Context common to all executions of this query
   "context": str = None,

   # List of data parallel items to instantiate the template with
   "data": [str] = None,

   # Number of times to execute the prompt (mutually exclusive with "data")
   n: int = None
}
```

Figure 4: Schema used to describe data-parallel tasks.

B Output Samples

Figure 5 presents an example of a raw, original query and Figure 6 demonstrates how this query can be adapted into the schema. Figures 7 and 8 present the serial and parallel outputs respectively for this query. Figure 3 then demonstrates the reasoning provided by `GPT-4o` for its automated quality evaluation.

```
original = \
"""Answer the questions about the given text:

Has the structure
     in the text been followed: 1. introduction 2. personal opinion 3. conclusion?
Have words and expressions
     in the text been used correctly and appropriately, both simple and more complex?
Are there any lexical errors in the text?
Have grammatical constructions in the text been used correctly?
Are there any grammatical errors in the text?
Are there any spelling errors in the text?
Are there any punctuation errors in the text?
What is the CEFR level of this text?
In my opinion, success is mostly related with
     hard work. If you start work on you, youll become better than you were yesterday
    . This fact is very simple, but on the other hand, it will grow a great person
     inside you. You shouldnt look to other people because the great enemy of yours is
     yourself. What is success? Is it having a will to get something or become someone
    ? I think, if you have an opportunity to buy something, it can not be called a
     success. But if you made that opporutniy from nothing, I can say that you succeed
    . But how did you get an opportunity? If you only depend on NAME_1 , it is possible
     to get some opportunity, but then nothing can use it but you, even your NAME_1.
     Hard work is basic. Due to hard work you can get anything you want. Maybe it takes
     much time , but the result would be sweet. NAME_1 is only a small percent of your
     success. I dont believe in talant from birth, only hard work.Many people think
    that money and success is equal, some poeple think that NAME_1 and success is equa
```

Figure 5: An example of an original query from the reading comprehension task.

```
{
   "serial": """Answer the questions about the given text:

Has the structure
     in the text been followed: 1. introduction 2. personal opinion 3. conclusion?
Have words and expressions
     in the text been used correctly and appropriately, both simple and more complex?
Are there any lexical errors in the text?
Have grammatical constructions in the text been used correctly?
Are there any grammatical errors in the text?
Are there any spelling errors in the text?
Are there any punctuation errors in the text?
What is the CEFR level of this text?

In my opinion, success is mostly related with
     hard work. If you start work on you, you'll become better than you were yesterday
    . This fact is very simple, but on the other hand, it will grow a great person
    inside you. You shouldn't look to other people because the great enemy of yours is
     yourself. What is success? Is it having a will to get something or become someone
    ? I think, if you have an opportunity to buy something, it can not be called a
    success. But if you made that opportunity from nothing, I can say that you succeed
    . But how did you get an opportunity? If you only depend on luck, it is possible
    to get some opportunity, but then nothing can use it but you, even your luck. Hard
    work is basic. Due to hard work you can get anything you want. Maybe it takes much
    time, but the result would be sweet. Luck is only a small percent of your success.
    I don't believe in talent from birth, only hard work. Many people think that money
     and success is equal, some people think that talent and success is equal.""",

"template": """Answer the following question about the given text:

{context}

Question: {data}""",

"context": """In my opinion, success is mostly related with
     hard work. If you start work on you, you'll become better than you were yesterday
    . This fact is very simple, but on the other hand, it will grow a great person
    inside you. You shouldn't look to other people because the great enemy of yours is
     yourself. What is success? Is it having a will to get something or become someone
    ? I think, if you have an opportunity to buy something, it can not be called a
    success. But if you made that opportunity from nothing, I can say that you succeed
    . But how did you get an opportunity? If you only depend on luck, it is possible
    to get some opportunity, but then nothing can use it but you, even your luck. Hard
    work is basic. Due to hard work you can get anything you want. Maybe it takes much
    time, but the result would be sweet. Luck is only a small percent of your success.
    I don't believe in talent from birth, only hard work. Many people think that money
     and success is equal, some people think that talent and success is equal.""",

"data": [
      "Has the structure
     in the text been followed: 1. introduction 2. personal opinion 3. conclusion?",
      "Have words and expressions
    in the text been used correctly and appropriately, both simple and more complex?",
      "Are there any lexical errors
     in the text?", "Have grammatical constructions in the text been used correctly?',
      "Are there any grammatical errors in the text?",
      "Are there any spelling errors in the text?",
      "Are there any punctuation errors in the text?",
      "What is the CEFR level of this text?"
   ]

}
```

Figure 6: The full schema generated from the example query presented in Figure 5.

```
**1. Has the structure
     in the text been followed: 1. introduction 2. personal opinion 3. conclusion?**
- Yes, the
     structure has been followed. The text starts with an introduction to the concept
     of success, presents the author's personal opinion on what constitutes success
    , and concludes with a reflection on the role of hard work versus luck and talent.

**2. Have words and expressions
    in the text been used correctly and appropriately, both simple and more complex?**
- Mostly,
     yes. The text uses a mix of simple and more complex expressions appropriately to
    convey the author's opinions on success. However, there are minor issues with the
    accuracy of some expressions, such as "related with" which should be "related to."

**3. Are there any lexical errors in the text?**
- Yes, there is a lexical error with "related with" which should be "related to."

**4. Have grammatical constructions in the text been used correctly?**
- Generally, yes, but there are some issues, such as the incorrect use of "but then
     nothing can use it but you," which is awkwardly phrased and could be clarified.

**5. Are there any grammatical errors in the text?**
- Yes, there are grammatical errors. For example, "If you start
    work on you" could be rephrased for clarity to "If you start working on yourself."

**6. Are there any spelling errors in the text?**
- No, there do not appear to be any spelling errors in the text.

**7. Are there any punctuation errors in the text?**
- Yes, there
     are a few punctuation errors. For instance, the use of commas is occasionally
     off, such as in "I think, if you have an opportunity to buy something, it can
     not be called a success," where the placement of the first comma is unnecessary.

**8. What is the CEFR level of this text?**
- Based on the complexity of the vocabulary and sentence structures, along
     with the errors present, this text could be placed around the B1-B2 level of the
     Common European Framework of Reference for Languages (CEFR). It demonstrates an
     intermediate level of language use, with the ability to convey personal opinions
     and abstract concepts, but with noticeable grammatical and lexical inaccuracies.
```

Figure 7: The serial output generated in response to the query given in Figure 5.

```
Yes, the structure
     in the text has been followed: 1. introduction 2. personal opinion 3. conclusion.

Yes, the words and
     expressions in the text have been used correctly and appropriately, encompassing
     both simple and more complex language to convey the author's viewpoint on success
    , its relation to hard work, and its distinction from luck, talent, and monetary
     wealth. The text effectively communicates the idea that success is primarily the
     result of hard work rather than external factors such as luck or innate talent.

Yes, there are lexical errors in the
    text. For example, "start work on you" should likely be "start working on yourself
    ," and "nothing can use it but you" could be more clearly expressed. Additionally
    , phrases like "made that opportunity from nothing" might be better as "created
     that opportunity from nothing," and expressions like "money and success is
    equal" should be "money and success are equal," to correct subject-verb agreement.

The text contains several grammatical
     errors and awkward constructions. Here are a few corrections and observations:

1. "related with" should be "related
     to" in the sentence "In my opinion, success is mostly related with hard work."
2. "If you start work on you," could be more clearly stated
     as "If you start working on yourself," for clarity and grammatical correctness.
3. The phrase "it will grow a great person inside you" could
     be rephrased for clarity, such as "it will help you grow into a great person."
4. The sentence "What is success? Is it having a will to get
     something or become someone?" could be streamlined for conciseness and clarity.
5. "I think
    , if you have an opportunity to buy something, it can not be called a success." The
     phrase "it can not" is more conventionally written as "it cannot" or "it can't."
6. In "But if you made
    that opportunity from nothing, I can say that you succeed," "succeed" should be in
     the past tense, "succeeded," to maintain consistency with the past tense "made."
7. "Due to hard work you can get anything
     you want." could be improved with a comma after "Due to hard work," for clarity.
8. The overall structure and punctuation of the text
     can be refined for better readability and coherence. For example, breaking down
    longer sentences into shorter, more concise ones could improve clarity and impact.

These adjustments would make the text grammatically correct and easier to understand.
```

```
Yes, there are several grammatical errors in the text. Here are a few examples:

1. "related with hard work" should be "related to hard work."
2. "If you
     start work on you" could be better phrased as "If you start working on yourself."
3. "This fact is very simple, but on the
     other hand, it will grow a great person inside you." could be clearer. A possible
    revision could be, "This concept is simple, yet it fosters great personal growth."
4. "You shouldn't look to other people because
     the great enemy of yours is yourself." could be rephrased for clarity, such as "
    You shouldn't compare yourself to others because your greatest enemy is yourself."
5. "But if you made that opportunity from nothing, I can say that you succeed." should
     be "But if you create that opportunity from nothing, then you have succeeded."
6. "But how did you get an opportunity?" could
    benefit from a smoother transition, like "But how does one obtain an opportunity?"
7. "nothing can use it but you, even your luck." could
    be clearer, perhaps "nothing but your own efforts, not even luck, can utilize it."
8. "Due to hard work you can get anything you want." could be more
     grammatically correct as "Due to hard work, you can achieve anything you want."
9. "Maybe it takes much time, but the result would
     be sweet." could be "It may take a long time, but the result will be rewarding."
10. The overall text could benefit from more precise
    punctuation and clearer sentence structures to improve readability and coherence.

Yes, there is a spelling error in the text. The phrase "start work
     on you" should likely be "start working on yourself" for clarity and grammatical
     correctness. Additionally, the text uses "depend on luck" and "nothing can use
     it but you," which might be considered awkward or imprecise phrasing rather than
     spelling errors. However, there are no conventional spelling mistakes in terms
     of incorrect letter sequences within words as presented in the original query.

Yes, there are punctuation errors in the text. For instance, commas
     are missing in several places where they are needed for clarity or grammatical
     correctness, such as after introductory phrases ("In my opinion," "But how
     did you get an opportunity?"). Additionally, the text could improve its clarity
     and readability with the use of commas or other punctuation in complex sentences.

The text provided does not strictly align
    with a specific Common European Framework of Reference for Languages (CEFR) level
    because it is not originally designed for language learning or assessment purposes
    . However, based on the complexity of the vocabulary, the sentence structure, and
     the conceptual depth of discussing success, hard work, and personal development
    , it could be roughly placed at a B2 level. This level is characterized by the
    ability to understand the main ideas of complex text on both concrete and abstract
     topics, including technical discussions in their field of specialization, and the
     capacity to interact with a degree of fluency and spontaneity that makes regular
     interaction with native speakers quite possible without strain for either party.
```

Figure 8: The parallel output generated in response to the query given in Figure 5. Note that the individual parallel generations are concatenated together.

```
Both responses accurately follow
     the prompt's instructions by addressing the required questions regarding the
     text's structure, word usage, lexical errors, grammatical correctness, spelling
    , punctuation, and CEFR level. They both identify similar issues within the text
    such as lexical and grammatical errors, thus scoring equally in terms of accuracy.

In terms of grammar, both responses are similar
     in terms of their grammatical construction and clarity. There are no significant
     differences between them that would affect their grammatical assessment.

Regarding detail, both responses provide similar levels of detail
    . They each identify key areas where the original text has linguistic issues and
     make specific suggestions for improvements, thus making them equally detailed.

Lastly, for preference, there is no clear standout between the two responses as they
     both provide equally valid analyses of the text and are written with comparable
     clarity and detail. Thus, there is no preferred response over the other.
```

Figure 9: The reasoning provided for an automated evaluation by GPT-4o for the responses given in Figures 7 and 8.