LLMEnsembleEval: A Modular Framework for Large Language Model Ensemble Evaluation

Beniamin Eryk Lobodziec

June 8, 2025

AALBORG UNIVERSITY
DENMARK

Department of Computer Science

AALBORG UNIVERSITY
STUDENT REPORT

Title:
LLMEnsembleEval: A Modular Framework for Large Language Model Ensemble Evaluation

Theme:
Master's thesis

Project Period:
10. Semester

Participant(s):
Beniamin Eryk Lobodziec

Supervisor:
Andres Masegosa

Copies: 1

Page Numbers: 25

Date of Completion:
June 8, 2025

Abstract:

Large Language Models (LLMs) achieve remarkable performance across diverse NLP tasks, yet suffer from critical reliability issues including hallucinations and inconsistent outputs. Ensemble methods emerge as promising solutions by combining predictions from multiple models to improve robustness and performance. However, current ensemble evaluation practices lack standardization, hindering method comparison and reproducibility.

This work addresses two key challenges in LLM ensemble research. First, we validate the Generation of Each token by LLMs as a Classification (GAC) strategy by reproducing core results and extending evaluation to additional models and benchmarks. Our experiments across MMLU, PIQA, ARC Challenge, and Winogrande reveal that GAC's effectiveness depends critically on performance similarity between ensemble members, with uniform weighting working best when models have comparable capabilities.

Second, we develop LLMEnsembleEval, the first standardized framework for LLM ensemble evaluation that integrates with lm-evaluation-harness. The modular architecture supports multi-GPU deployment and enables systematic comparison of ensemble strategies while maintaining reproducible protocols.

Our findings demonstrate that GAC consistently improves performance on knowledge-intensive tasks like MMLU (gains of 0,1% to 3,6%) but shows mixed results on complex reasoning tasks, highlighting the need for task-specific strategies. The performance similarity hypothesis show that ensembles work best with models of comparable capability.

LLMEnsembleEval provides the foundation for systematic evaluation of emerging ensemble strategies, potentially accelerating progress toward more reliable and effective LLM systems.

Summary

Large Language Models (LLMs) have revolutionized natural language processing but suffer from critical reliability issues including hallucinations, inconsistent outputs, and brittle behavior. Ensemble methods have emerged as a promising solution by leveraging collective intelligence of multiple models to improve robustness and performance. However, current ensemble evaluation practices lack standardization, making it difficult to compare methods or reproduce findings across studies.

This project addresses two key challenges. First, we validate the Generation of Each token by LLMs as a Classification (GAC) strategy by reproducing its core results and extending evaluation to additional models and benchmarks. Second, we develop LLMEnsembleEval, a standardized framework that integrates ensemble evaluation with the widely-adopted lm-evaluation-harness, enabling systematic comparison of ensemble methods.

Our key contributions include successfully reproducing the original GAC results with an average difference of ±0,675% on MMLU and extending evaluation to seven models across four benchmarks (MMLU, PIQA, ARC Challenge, and Winogrande). Through systematic analysis, we discover that GAC's effectiveness depends critically on performance similarity between ensemble members. Models with comparable capabilities benefit more from uniform weighting than combinations with large performance gaps. We also develop the first standardized library for LLM ensemble evaluation, featuring multi-GPU support, vocabulary unification across different tokenizers, and seamless integration with existing evaluation frameworks.

Our experimental results reveal important patterns. GAC consistently improves performance on knowledge-intensive tasks like MMLU (gains of 0,1% to 3,6%) but shows mixed results on complex reasoning tasks. The most significant finding is the performance similarity hypothesis: ensembles combining models with similar baseline performance consistently show improvements, while ensembles with large performance gaps typically underperform. This suggests that uniform weighting dilutes stronger model predictions with noise from weaker models, limiting beneficial diversity.

These findings have important implications for ensemble design. Rather than maximizing model diversity. The limitations of static uniform weighting point toward adaptive ensemble strategies that adjust weights based on model confidence and task characteristics. The LLMEnsembleEval framework provides the foundation for systematic evaluation of emerging ensemble strategies, potentially accelerating progress towards more reliable and effective LLM systems.

Contents

1 Introduction **1**

1.1 Research Objectives . . . 2

2 Related Work **3**

2.1 Ensemble Strategies for Large Language Models . . . 3

2.1.1 LLM's Ensemble Strategies . . . 3

2.2 Evaluation of Large Language Models . . . 4

2.2.1 Popular Benchmarks . . . 4

2.2.2 Evaluation Metrics . . . 5

2.2.3 lm-evaluation-harness Framework . . . 6

2.3 Limitations in Evaluating Ensembles of LLMs . . . 6

2.4 Inconsistent Use of Models and Benchmarks . . . 6

2.4.1 Cherry-picking and Reporting Bias . . . 7

2.4.2 Need for Standardized Evaluation . . . 7

3 Methodology **8**

3.1 Overview of GAC . . . 8

3.2 LLM Evaluation Framework . . . 9

4 LLMEnsembleEval: A Multi-GPU LLM Ensemble Framework **11**

4.1 Multi-GPU Architecture and Implementation . . . 12

4.2 Ensemble Class . . . 12

4.3 Wrapper Class . . . 13

5 Results **15**

5.1 Implementation Validation: Reproducing Original Results . . . 15

5.2 Extended Evaluation: Demonstrating GAC Effectiveness . . . 15

5.3 Benchmark Specific Analysis . . . 16

5.3.1 MMLU Performance . . . 16

5.3.2 PIQA Performance . . . 16

5.3.3 ARC Challenge Performance 17
5.3.4 Winogrande Performance 17
5.4 Performance Patterns 17

6 Discussion **18**
6.1 Key Findings 18
6.2 Model Diversity and Ensemble Error Analysis 18
6.3 Limitations 19

7 Conclusion **20**
7.1 Key Contributions 20
7.2 Future Directions 20

8 Appendix **21**

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing, achieving performance improvements across diverse tasks from text generation to complex reasoning [3] [9] [11]. These models, trained on large amounts of text data, have achieved high performance on numerous benchmarks and have become important in both research and practical applications. Their ability to understand context, generate coherent responses and adapt to diverse tasks has positioned them as foundational technologies in artificial intelligence.

Despite their capabilities, LLMs exhibit several critical limitations that hinder their reliable deployment in real-world applications. These models are prone to hallucinations, generating plausible sounding but factually incorrect information, and demonstrate brittle behavior where small changes in input can lead to different outputs.

Ensemble methods for LLMs have emerged as a promising approach to address these issues by leveraging collective intelligence of multiple models. Research has demonstrated that combining predictions from multiple LLMs can improve robustness, reduce hallucinations and enhance the overall performance across various benchmarks [4] [14]. These strategies aggregate diverse model outputs using various algorithms. The underlying idea behind ensemble is that while individual models may fail in different ways, their combined response can compensate for their individual weakness and provide more reliable results. Among these approaches, the Generation of Each token by LLMs as a Classification (GAC) strategy has emerged as one of the more recent and effective ensemble methods for LLMs, demonstrating performance improvements across multiple benchmarks[17].

However, current evaluation practices for LLM ensembles lack standardization and reproducibility, which significantly limits the fields progress. Most ensemble research implement custom evaluation approaches rather than adopting existing frameworks like lm-evaluation-harness [13], making it difficult to compare results across studies or reproduce findings. Papers frequently report benchmark findings without detailing how ensemble outputs were scored and normalized, creating a barrier to meaningful scientific comparison. While GAC explicitly reports using the lm-evaluation-harness for its experiments, even reproducing the results remains challenging due to the complexity of coordinating multiple models and integrating them with the standardized evaluation framework, which is designed to be used on single models from the Huggingface website. There is therefore a need for standardized software library that simplifies the process of implementation and evaluation of LLM ensembles, enabling researchers to focus on developing novel ensemble approaches rather than building the evaluation infrastructure from scratch.

1.1 Research Objectives

This project has two primary objectives:

- Validate the Generation of Each token by LLMs as a Classification ensembling strategy by reproducing its core results using same models and evaluation tool. By replicating the performance gains across established benchmarks, this project aims to assess the robustness and generalizability of the original Generation of Each token by LLMs as a Classification approach.
- Develop a practical and reusable implementation that simplifies the process of ensembling and evaluating Large Language Models (LLMs). This involves integrating the ensemble workflow with the lm-evaluation-harness framework. Enabling researchers to more easily adopt and benchmark ensemble strategies without building custom implementation from scratch.

2 Related Work

Recent advances in Large Language Models (LLMs) have inspired growing research on ensemble strategies that aim to combine the strengths of multiple models. This chapter reviews recent ensemble strategies for Large Language Models (LLMs), focusing on methods that combine token-level outputs from multiple models. Additionally, we examine how these ensemble methods are typically evaluated, highlighting current limitations in standardization and reproducibility. This review concludes by identifying key research gaps that motivate the work presented in this thesis.

2.1 Ensemble Strategies for Large Language Models

Ensemble methods for LLMs operate by combining outputs from multiple models to improve performance, robustness and reliability. Beyond the performance improvements, these approaches also target LLM limitations including hallucinations, inconsistent outputs and brittle behavior when answering complex tasks. Figure 1 illustrates the general concept of LLM ensembling.

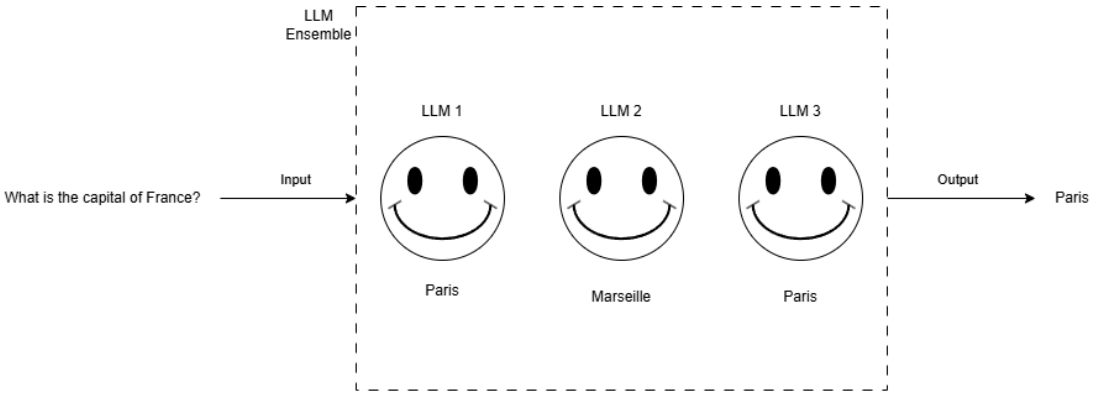

Figure 1: Overview of LLM Ensembles.

2.1.1 LLM's Ensemble Strategies

Many different ensemble approaches have been proposed, each with different strategy for combining model outputs:

Generation of each token by LLMs as a Classification (GAC) treats each token prediction as a classification task, combining probability distributions from multiple models to calculate the next token [17]. At each generation step, every model independently computes probabilities over its vocabulary. These distributions are then aggregated through weighted averaging to create a unified probability distribution for token selection.

UNIon Top-k Ensembling (UNITE) focuses only on the top-k most likely tokens from each model rather than aligning entire vocabularies. This reduces computational processing while respecting each model's unique tokenization [4]. This approach creates a union of the most promising candidates across all models, then renormalizes probabilities over this reduced set.

SWEETSPAN operates at the span level, generating text spans from multiple models which are then evaluated and selected based on quality metrics [14]. Each model candidate is generated on a shared prefix, then perplexity scores are calculated. This approach balances the need for real-time adjustments with the information required for accurate ensemble decisions.

Ensemble LLMs via Vocabulary Alignment (EVA) creates mappings between different model vocabularies using overlapping tokens and uses semantic similarity to map unique tokens. This approach allows for a projection to a unified vocabulary space for ensemble aggregation [15].

DeepEn fuses probability distributions for different LLMs at each decoding step [7]. The method handles vocabulary differences by mapping distributions to a shared space, performing aggregation, then converting back to select the next token.

2.2 Evaluation of Large Language Models

Evaluating the performance of LLMs involves a combination of standardized benchmarks, performance metrics and software tools designed to ensure compatibility and reproducibility. Most commonly, LLMs are assessed on tasks that test reasoning, factual knowledge and commonsense understanding.

2.2.1 Popular Benchmarks

Several benchmarks have emerged as standard tools to assess different capabilities of LLMs:

- **MMLU** tests knowledge across 57 academic subjects in a multiple-choice format [6].

  Example:
  ```
  Question: What is the capital of Canada?
  Choices: (A) Toronto (B) Montreal (C) Ottawa (D) Vancouver
  Answer: C
  ```

- **GSM8K** evaluates multi-step mathematical reasoning problems [5].

  Example:
  ```
  Question: If you buy 3 apples for $2 each and 2 bananas for $1 each,
  how much did you spend?
  Answer: 8
  ```

- **PIQA** tests physical commonsense understanding [2].

  Example:
  ```
  Goal: To break a glass bottle...
  Choice A: Throw it against a wall
  Choice B: Wrap it in a towel
  Correct: A
  ```

- **TriviaQA** focuses on factual question answering using open-domain knowledge [8].

  Example:
  ```
  Question: Who discovered penicillin?
  Answer: Alexander Fleming
  ```

- **ARC (AI2 Reasoning Challenge)** and **Winogrande** test scientific and commonsense reasoning, respectively [16, 12].

  Winogrande Example:
  ```
  Sentence: The trophy doesn't fit into the suitcase because it is
  too small.
  Question: What is too small? (trophy/suitcase)
  Answer: suitcase
  ```

These benchmarks vary in complexity, format and the type of reasoning they require from the LLM, which makes them useful for stress-testing different aspects of LLM behavior.

2.2.2 Evaluation Metrics

Different tasks require different metrics, but the most commonly used ones include:

- **Accuracy:** Measures the percentage of correct answers in classification or multiple-choice tasks. It is widely used in benchmarks.

- **Perplexity:** Used in language modeling tasks, perplexity quantifies how well a model predicts the next word in a sequence. Lower perplexity indicates a better understanding of language structure and context.

- **Exact Match (EM):** Used for tasks with a single correct textual output. EM checks whether the model's response exactly matches the reference answer.

- **F1 Score:** Balances precision and recall, especially useful when model outputs are partial matches to the expected answer.

- **Log-likelihood:** Captures the probability assigned by a model to a sequence of tokens, making it particularly suitable for comparing model confidence or aggregating token-level outputs in ensembles.

These metrics are task-dependent and often used together to provide more comprehensive assessment of model's performance.

2.2.3 lm-evaluation-harness Framework

One of the most widely adopted open-source frameworks for evaluating large language models is the **lm-evaluation-harness**, developed by EleutherAI. This tool provides standardized implementations of dataset loading, few-shot prompt formatting, model inference, and metric computation for over 40 popular NLP benchmarks, including MMLU, PIQA, GSM8K, and HellaSwag.

The framework is tightly integrated with the Hugging Face ecosystem and was notably used to power the Open LLM Leaderboard[1], which publicly ranked the performance of open-weight models across multiple standardized tasks. Although the leaderboard is now archived, its reliance on **lm-evaluation-harness** helped establish the tool as the standard for reproducible LLM evaluation in the open-source community.

By default, **lm-evaluation-harness** supports evaluation of single models hosted on Hugging Face. However, its flexible architecture allows for integration of custom models, including ensemble methods, via a wrapper class. This extensibility is leveraged in this project to enable ensemble evaluation, even though native support for multi-model coordination and aggregation logic is not provided. As a result, meaningful ensemble evaluation still requires additional engineering to conform to the framework's expected interfaces.

2.3 Limitations in Evaluating Ensembles of LLMs

Although the evaluation of single LLMs has matured significantly, the assessment of LLM ensembles remains underdeveloped. Current literature on ensemble methods often lacks transparency, standardization, and consistency in experimental setups.

2.4 Inconsistent Use of Models and Benchmarks

A major limitation in ensemble research is the lack of common evaluation setups. Different studies use different:

- Model combinations (which vary in architecture and size)
- Benchmarks (often cherry-picked to favor the method and showcase improvements.)
- Metrics or task-specific adaptions (which are often undocumented)

This inconsistency severely limits the ability to compare results across papers or to draw generalized conclusions about the effectiveness of ensemble strategies.

For example, while some papers evaluate their ensemble models on MMLU, others use completely different benchmarks, with little justifications for these choices. Moreover, many papers do not mention critical implementation details about scoring, normalization or token alignment, making it difficult reproducing the results.

2.4.1 *Cherry-picking and Reporting Bias*

Because ensemble performance often depends heavily on the combination of models used, some papers selectively report only the most favorable results. This introduces the cherry-picking bias, where gains are overstated and potential weakness is hidden. In this project, we later explore a wide range of model combinations to observe the ensemble effectiveness. These variations showcase the importance of evaluating ensemble strategies comprehensively rather than selectively.

2.4.2 *Need for Standardized Evaluation*

While flexible frameworks like **lm-evaluation-harness** exist, they are rarely adopted in ensemble research. Even when they are used such as the GAC paper [17]. Their native support for ensemble evaluation is limited. This requires significant engineering to adapt the framework for multi-model coordination, shared token handling and ensemble-specific scoring logic.

In this project, we address these issues by implementing an extensible ensemble evaluation pipeline within the **lm-evaluation-harness**. This setup enables fair, consistent comparison of ensemble strategies (such as GAC, UNITE, SWEETSPAN, and others) across standardized benchmarks and model combinations.

3 Methodology

This section presents the ensemble approach and evaluation methodology used in this project. We first describe the GAC strategy in our ensemble implementation. We then outline the **lm-evaluation-harness** framework, which provides a standardized platform for evaluating model performance across multiple benchmarks.

3.1 Overview of GAC

The GAC ensemble strategy treats token generation as a classification problem where multiple models contribute to predicting the next token at each step [17].

Token-Level Classification: At each generation step, GAC treats the prediction of the next token as a classification task. Each model in the ensemble independently computes the probability distribution over its entire vocabulary, representing the likelihood of each possible token being the correct next token given the current context.

Probability Aggregation: The core of GAC lies in combining these individual probability distributions. For models sharing the same tokenizer (such as models from the same family), probabilities can be directly averaged across the vocabulary positions. This aggregation process creates a unified probability distribution that incorporates the collective knowledge of all ensemble members.

Handling Vocabulary Differences: When ensemble models use different tokenizers or vocabularies, GAC faces the challenge of aligning probability distributions across incompatible token spaces. The method addresses this by implementing vocabulary mapping strategies to enable meaningful probability combination.

Sequential Generation: Once the aggregated probability distribution is computed, GAC selects the token with the highest combined probability. This token is added to the current sequence, and the process repeats for the next position. Each model then uses the updated sequence to generate probability distributions for the next token prediction.

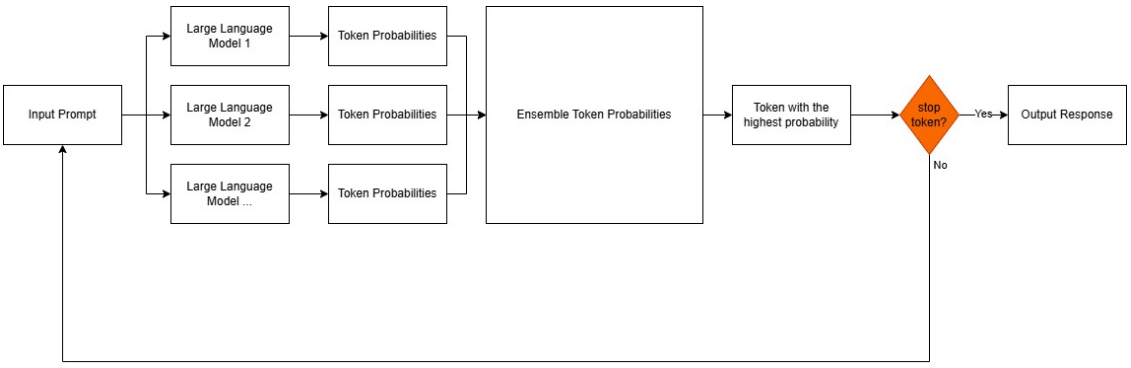

Figure 2: Overview of the GAC strategy.

3.2 LLM Evaluation Framework

The lm-evaluation-harness framework provides standardized implementations for evaluating language models across numerous benchmarks [13]. The framework is integrated with the Hugging Face ecosystem, enabling evaluation of models hosted on Hugging Face.

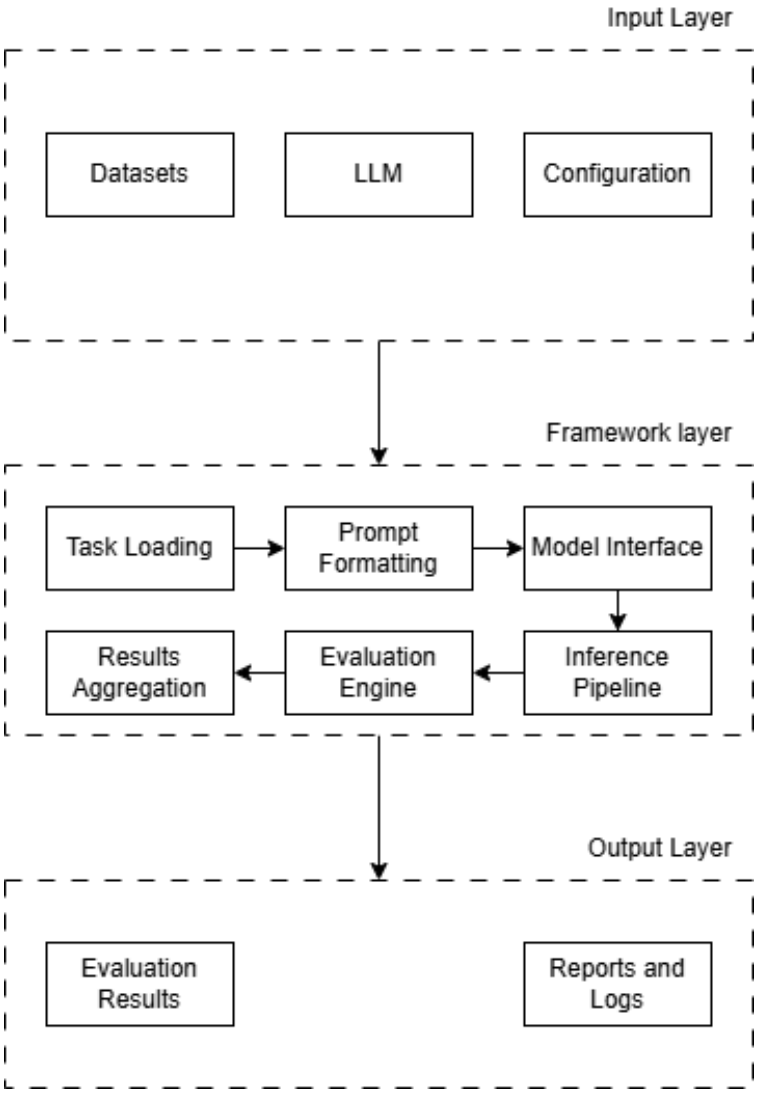

Figure 3: Overview of the framework architecture.

As shown in the figure 3 the framework coordinates several key components to provide standardized evaluation.

Inputs: The framework accepts datasets (such as MMLU, GSM8K and PIQA), LLM models (from huggingface or custom implementations), and configuration parameters (few-shot settings and batch sizes).

Core Processing: The framework handles task loading, prompt formatting and model interface management for tokenization and inference. The evaluation engine computes metrics while the inference pipeline

processes requests, with results aggregation normalizing scores across different tasks.

Outputs: The framework produces evaluation results and comprehensive reports with logs that ensure reproducible configurations.

Key features include:

- **Standardized benchmarks**: Implementations of popular datasets (MMLU, GSM8K, HellaSwag, etc.)
- **Consistent evaluation**: Uniform scoring procedures across different tasks
- **Reproducible results**: Standardized experimental setups that enable fair comparisons
- **Pre-configured model families**: Built-in support for various model architectures (GPT, BERT, LLAMA, etc.)
- **Extensible design**: Architecture that allows adding new benchmarks and models from beyond Huggingface

Its pre-configured support for different model families significantly simplifies the evaluation process. Users can evaluate models with minimal setup, as the framework handles model-specific configurations, tokenization, and inference procedures automatically. However, when working with custom models, we must implement specific methods in a so-called wrapper class to ensure compatibility with the framework's expected interfaces and achieve the same level of functionality as the pre-configured models. Furthermore, the framework was also used as a foundation for the now archived Open LLM Leaderboard on huggingface. [1]

While lm-evaluation-harness excels at evaluating individual models, it has inherent limitations when working with ensembles. The framework is designed around single-model evaluation and lacks native support for ensemble coordination, aggregation logic, or multi-model response handling. The custom model feature allows for ensemble integration, but requires implementing the necessary methods to represent ensemble outputs as single model responses that respects the framework's evaluation expectations.

By addressing the inconsistency problem in ensemble evaluation, the framework offers a reproducible foundation for benchmarking. This allows researchers to focus on developing ensemble strategies rather than building evaluation infrastructure from scratch.

4 LLMEnsembleEval: A Multi-GPU LLM Ensemble Framework

To evaluate the GAC ensemble using the **lm-evaluation-harness** framework, we have developed LLMEnsembleEval, a comprehensive library designed around a modular architecture built around two main components: the **Ensemble** class and the **Wrapper** class. This separation ensures flexibility, reusability, and seamless integration with existing Hugging Face workflows while providing robust multi-GPU support for efficient ensemble processing.

Figure 4 provides a conceptual overview of our library architecture, illustrating how multiple LLMs are distributed across GPUS and integrated with the lm-evaluation-harness for standardized evaluation.

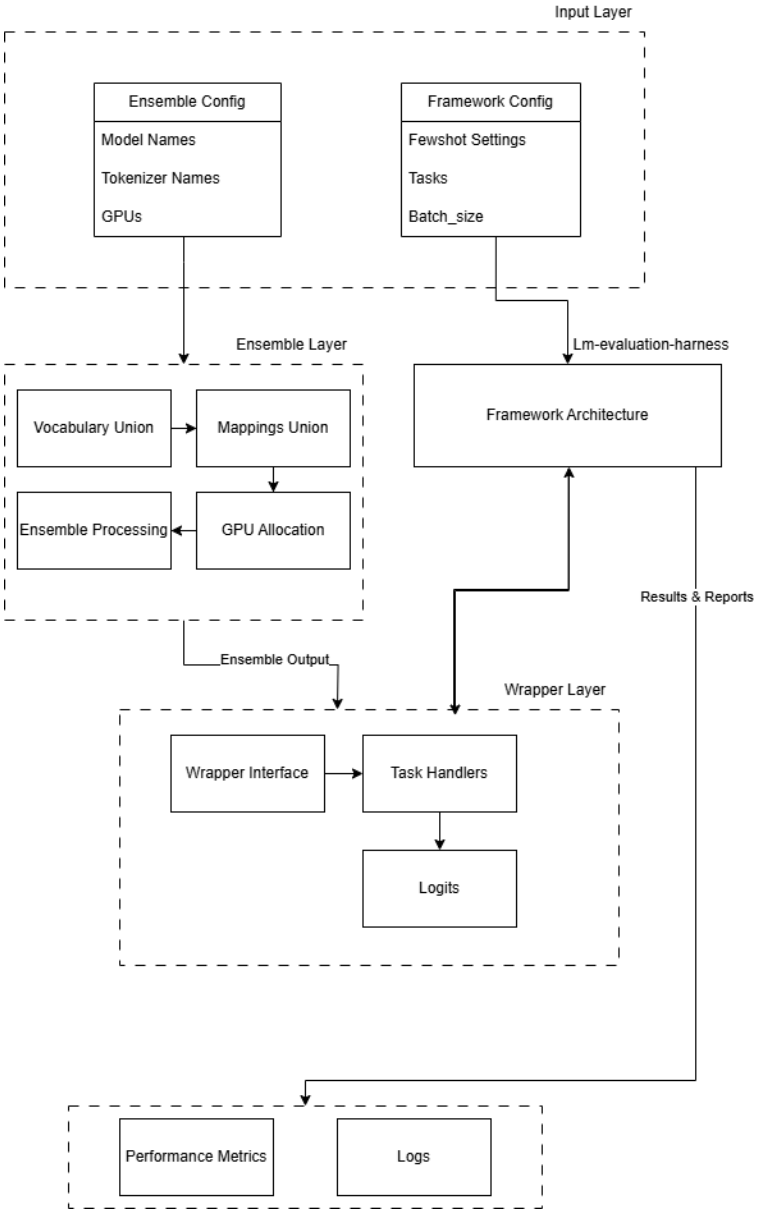

Figure 4: Overview of the conceptual LLMEnsembleEval architecture.

The complete implementation of the library, including source code, is available on github: https://github.com/blobod/LLMEnsembleEval.

4.1 Multi-GPU Architecture and Implementation

LLMEnsembleEval supports multi-GPU deployment to address the memory and computational requirements of loading multiple large language models simultaneously. The framework automatically distributes models across available CUDA devices, enabling parallel inference while maintaining memory efficiency. This architecture allows to experiment with larger model combinations that would exceed single-GPU memory limits.

4.2 Ensemble Class

The Ensemble class contains the core logic for combining predictions from multiple Hugging Face models at the token level. It handles three critical responsibilities: vocabulary unification, logit aggregation, and maintaining Hugging Face compatibility.

Vocabulary Unification: The most challenging aspect of ensemble implementation is integrate different tokenization schemes across models. For example, Qwen-1.5 uses "_" as a whitespace prefix while Phi-3-mini uses "Ġ". Without proper handling, tokens like "_Answer" from one model would not align with "ĠAnswer" from another, causing ensemble failures.

The solution involves constructing a union vocabulary across all models and creating mapping tables from each model's vocabulary to this shared space. The system detects the dominant whitespace prefix among ensemble members and normalizes all tokens accordingly:

```
def _create_vocab(self):
    """Create a UNION vocabulary with special handling for different tokenizer types."""
    ...
    # Detect dominant prefix (e.g., G or _)
    ...
    # Normalize tokens using dominant prefix
    ...
    # Add normalized tokens to union vocab
    ...
```

Listing 1: Create vocab method

The complete method implementation is provided in the Appendix 3.

Hugging Face Integration: The ensemble implements standard Hugging Face methods such as(**generate**, **save_pretrained**, **from_pretrained**), allowing it to be used as a drop-in replacement for any individual model in existing workflows.

4.3 Wrapper Class

The Wrapper class serves as an adapter between the Ensemble and lm-evaluation-harness, implementing the evaluation framework's expected interface methods (**loglikelihood**, **loglikelihood_rolling**, **generate_until**). This design allows the ensemble to be evaluated using the same protocols as individual models.

Interface Implementation: The wrapper retrieves token-level probabilities from the ensemble, processes them according to harness requirements, and returns properly formatted results. For instance, the **loglikelihood** method aggregates probabilities across the union vocabulary and converts them to the log-likelihoods expected by the evaluation framework.

```
def loglikelihood(self, requests):
        """Calculate log-likelihood for a batch of requests."""
        if not requests:
            return []

        results = []

        for idx, instance in enumerate(requests):
            try:
                # Validate instance structure
                if not (hasattr(instance, 'args') and isinstance(instance.args, tuple) and
                    len(instance.args) == 2):
                    results.append((-float('inf'), False))
                    continue
                context, continuation = instance.args
                doc = getattr(instance, 'doc', {})
                # Detect benchmark type and call appropriate handler
                detected_type = self._detect_benchmark_from_instance(instance, debug=False)
                if detected_type == "piqa":
                    log_prob, is_greedy = self._handle_piqa(context, continuation, doc,
                        debug=True)
                elif detected_type == "mmlu":
                    log_prob, is_greedy = self._handle_mmlu(context, continuation, doc,
                        debug=True)
                elif detected_type == "arc":
                    log_prob, is_greedy = self._handle_arc(context, continuation, doc, debug
                        =True)
                elif detected_type == "winogrande":
                    log_prob, is_greedy = self._handle_winogrande(context, continuation, doc
                        , debug=True)
                else:
                    # Raise error for unknown benchmark types to ensure proper handling
                    raise ValueError(f"Unknown benchmark type detected: {detected_type}. "
                                     f"Supported benchmarks: piqa, mmlu, arc and winogrande")
```

```
30
31                results.append((float(log_prob), is_greedy))
32
33            except Exception as e:
34                # Log error for debugging but continue processing
35                print(f"Error processing instance -idx : -e ")
36                results.append((-float('inf'), False))
37
38        return results
```

Listing 2: Wrapper loglikelihood implementation

Benchmark-Specific Processing: Different evaluation tasks require specialized handling due to their unique output formats and scoring methods. The wrapper implements task-specific processors for the benchmarks used in this study:

- `_handle_mmlu`: Aggregates probabilities for multiple-choice answers across differently-tokenized outputs
- `_handle_piqa`: Manages log-probability summing for multi-token completions
- `_handle_lambada`: Performs text cleaning and target token alignment

Extensibility: The modular design supports other ensemble strategies with minimal modification. Any ensemble that implements the Hugging Face interface can utilize the existing wrapper, making the framework broadly applicable beyond GAC.

5 Results

In this section we present our experimental findings in two parts: first, we validate our implementation by reproducing the original paper's MMLU results using their exact models, then we demonstrate the broader effectiveness of GAC by extending the evaluation to additional models and benchmarks.

5.1 Implementation Validation: Reproducing Original Results

To establish the credibility of our GAC implementation, we will first reproduce key results from the original paper using their exact model combinations on MMLU with the same few-shot settings. This validation ensures our ensemble framework correctly implements the GAC algorithm. For this, we chose 4 models: **Qwen1.5-14B-Chat**, **Phi-3-mini-4k-instruct**, **openchat_3.5** and **Nous-Hermes-2-SOLAR-10.7B**

MMLU Reproduction Results:

Model Combination	Original Paper	Our Implementation	Difference
Qwen1.5-14B + Phi-3-mini	69,91%	71%	±1,09%
Nous-Hermes + openchat_3.5	66,51%	66,77%	±0,26%

Table 1: Direct comparison with original paper results on MMLU

Individual Model Baselines:

Model	Original Paper	Our Measurement	Difference
Qwen1.5-14B-Chat	67,2%	66%	±1,2%
Phi-3-mini-4k-instruct	67,11%	70%	±2,89%
openchat_3.5	63,87%	63,67%	±0,2%
Nous-Hermes-2-SOLAR-10.7B	64,88%	65,4%	±0,52%

Table 2: Individual model performance on MMLU

Validation Summary: Our implementation achieves reasonable alignment with the original findings, with an average difference of [±0,675%]. These minor variations could be the result of different handling of models within our wrapper class.

5.2 Extended Evaluation: Demonstrating GAC Effectiveness

Having validated our implementation, we will now extend the evaluation to demonstrate that GAC improves performance across multiple models and benchmarks.

Complete Model Set:

- **Original Paper Models**: Qwen1.5-14B-Chat, Phi-3-mini-4k-instruct, openchat_3.5 and Nous-Hermes-2-SOLAR-10.7B
- **Additional Models**: Qwen3-1.7B-Base, Llama-3.2-1B-Instruct and gpt2-medium

- **Additional Benchmarks:** PIQA[2], Arc Challenge[16] and Winogrande[12]

Individual Model Performance Across All Benchmarks:

Model	MMLU	PIQA	Arc Challenge	Winogrande
Qwen1.5-14B-Chat	66%	75,4%	44,8%	68,9%
Phi-3-mini-4k-instruct	70%	79,9%	53,8%	73,4%
openchat_3.5	61,6%	81,5%	57,7%	64,4%
Nous-Hermes-2-SOLAR-10.7B	63,8%	81,6%	57,6%	78,1%
Qwen3-1.7B-Base	62,6%	75,8%	41,8%	64,4%
Llama-3.2-1B-Instruct	45,5%	74,4%	35,7%	54,3%
gpt2-medium	26,49%	67,7%	21,5%	53,1%

Table 3: Baseline performance of the individual models across all benchmarks

GAC Ensemble Performance: All ensemble combinations use uniform 50-50 weighting between the two models.

Ensemble Combination	MMLU	PIQA	Arc Challenge	Winogrande
Qwen1.5-14B + Phi-3-mini	71% **[+1%]**	79,5% [-0,4%]	50,5% [-3,3%]	74% **[+0,6%]**
Qwen1.5-14B + openchat_3.5	68,8% **[+2,8%]**	80% [-1,5%]	50,8% [-6,9%]	71,9% **[+3%]**
Qwen1.5-14B + Nous-Hermes	69,6% **[+3,6%]**	80,6% [-1%]	50,9% [-6,7%]	74,7% [-3,2%]
Phi-3-mini + openchat_3.5	69,9% [-0,1%]	82,5% **[+1%]**	58,3% **[+0,6%]**	77,9% **[+4,5%]**
Phi-3-mini + Nous-Hermes	70,8% **[+0,8%]**	82,3% **[+0,7%]**	58,4% **[+0,8%]**	78,6% **[+0,5%]**
Nous-Hermes + openchat_3.5	66,7% **[+2,9%]**	82,2% **[+0,6%]**	56,6% [-1,1%]	77,9% [-0,2%]
Qwen3-1.7B + Qwen1.5-14B	68,6% **[+2,6%]**	77% **[+1,2%]**	45% **[+0,2%]**	69,7% **[+0,8%]**
Qwen3-1.7B + Llama-3.2-1B	61% [-1,6%]	76% **[+0,2%]**	41,3% [-0,5%]	64,7% **[+0,3%]**
gpt2-medium + Phi-3-mini	70,1% **[+0,1%]**	77,3% [-2,6%]	41,3% [-12,5%]	72,5% [-0,9%]

Table 4: GAC ensemble results across all tested combinations. Values in the bracket show the difference compared to the best performing individual model in each ensemble.

5.3 Benchmark Specific Analysis

5.3.1 MMLU Performance

The GAC strategy consistently improves performance on MMLU across nearly all tested model combinations, with gains ranging from 0,1% to 3,6% over the best individual model. This suggest particular effectiveness for knowledge intensive, multiple-choice tasks where the ensemble can leverage the diverse knowledge bases of different models.

5.3.2 PIQA Performance

Results on PIQA show mixed effectiveness. While some combinations achieve modest improvements (+0,2% to +1,2%), others show minor degradation (-0,4% to -2,6%). The commonsense reasoning required by PIQA appears to be more sensitive to the specific model combinations used. This suggests that the GAC

token-level averaging may not always capture the nuanced reasoning patters needed for these tasks.

5.3.3 ARC Challenge Performance

ARC Challenge results reveal more pronounced limitations of GAC. Most combinations under perform compared to their best individual model. The reason for this could be that the scientific reasoning tasks within ARC Challenge were already challenging, as most of the individual models achieved below 60% accuracy. The added complexity of token-level averaging appears to introduce more noise rather than complementary knowledge which results in most of the combinations having lower results.

5.3.4 Winogrande Performance

Winogrande shows that most GAC combinations provided improvements over their base models. Notable gains include +4,5% for Phi-3-mini + openchat_3.5 and +3% for Qwen1.5-14B + openchat_3.5, with 7 out of 9 combinations showing positive results. The few cases of lower results suggest that while GAC is generally effective for commonsense reasoning tasks, the compatibility of the models' reasoning approaches still plays a role in the success of the ensemble.

5.4 Performance Patterns

The results reveal several observable patterns across benchmarks:

- Ensembles with similar baseline performances (e.g., Phi-3-mini + openchat_3.5) generally show more consistent improvements
- Combinations with large performance gaps (e.g., gpt2-medium + Phi-3-mini) exhibit greater underperformance
- Knowledge-intensive tasks (MMLU) show more consistent ensemble benefits than complex reasoning tasks (ARC Challenge)

The implications of these patterns for ensemble design and the underlying causes of performance variations are analyzed in detail in Discussion. 6

6 Discussion

6.1 Key Findings

Implementation Validation: Our GAC implementation successfully reproduced the original paper's results with an average difference of $\pm 0,675\%$ on MMLU, demonstrating the correctness of our ensemble framework. Minor variations likely come from differences in model handling within our wrapper class.

Performance Similarity Hypothesis: Building on the performance patterns observed in section 5.4, our results suggest that GAC ensembles with uniform weights work most effectively when individual model performances are similar.

When models have different capabilities, uniform averaging dilutes the stronger models prediction with noise from the weaker model. In contrast, models with similar performance are more likely to make complementary errors that can be corrected through ensemble averaging.

6.2 Model Diversity and Ensemble Error Analysis

Our findings align with the diversity-error framework established by Ortega et al. [10], which demonstrates that ensemble success requires both low individual error and beneficial diversity among predictors. However, our results extend this theory to the LLM domain, revealing task-specific patterns not previously explored.

This relationship explains our mixed results. The theory demonstrates that for ensemble success, we need both low average individual error and high beneficial diversity, where models make different types of errors that can complement each other.

Our results suggest that when models have very different capability levels (gpt2-medium + Phi-3-mini), the diversity may not be beneficial. The weaker model introduces more noise than complementary information. As shown in their work, diversity decreases when predictors are highly correlated. We extend this by showing that excessive performance gaps can also limit beneficial diversity.

In contrast, models with similar performance levels (Phi-3-mini + openchat_3.5) can achieve beneficial diversity while maintaining low average error, aligning with the optimal balance described in the paper.

6.3 Limitations

Benchmark-Specific Performance: The inconsistent performance across different benchmarks highlights a key limitation of GAC. While effective for tasks like MMLU, the strategy shows mixed results for commonsense reasoning (PIQA) and struggles with complex reasoning tasks like ARC Challenge. This suggests that ensemble strategies may need to be tailored for specific task types and not something that can be applied universally.

Computational Overhead: Our implementation requires loading and maintaining multiple models simultaneously, which significantly increases memory requirements and inference time. The overall focus was on the correctness rather than efficiency.

Uniform Weighting as a Fundamental Constraint Beyond the computational overhead already discussed, our results reveal fundamental limitations of GAC's static 50-50 weighting approach that cannot adapt to varying model capabilities or confidence levels. This becomes increasingly problematic as performance disparities between ensemble members grow.

7 Conclusion

This report advances LLM ensembles research through empirical validation of ensemble theory and development of standardized evaluation tool. Our work demonstrates that GAC's effectiveness depends critically on model similarity, with uniform weights working best when ensemble members have comparable capabilities rather than requiring maximum diversity.

7.1 Key Contributions

We provide a comprehensive validation of the GAC strategy across multiple benchmarks, revealing strong performance on knowledge-intensive tasks (MMLU) while highlighting limitations in complex reasoning scenarios (Arc Challenge). Our findings demonstrate that performance similarity between ensemble members significantly affects GAC effectiveness.

LLMEnsembleEval addresses a critical gap by providing the first standardized framework for LLM ensemble evaluation. The modular design enables systematic comparison of ensemble methods while maintaining reproducible evaluation protocols.

7.2 Future Directions

The limitations of static weights revealed in this work point towards adaptive ensemble strategies that consider model confidence and task characteristics. Future research should explore performance clustering for ensembles and develop methods that would distinguish beneficial diversity from the harmful one, across different task types.

Our framework provides the foundation for systematic evaluation of emerging ensemble strategies like UNITE, SWEETSPAN and EVA, potentially accelerating progress towards more reliable and effective LLM ensemble strategies.

8 Appendix

```
    def _create_vocab(self):
            """Create a UNION vocabulary with special handling for different tokenizer types
                .."""
            print("[GAC _create_vocab] Creating improved UNION vocabulary...")

            # Initialize for the union vocabulary
            self.union_vocab_set = set()

            # Process each tokenizer to find dominant prefix type
            g_prefix_count = 0
            u_prefix_count = 0

            for tokenizer in self.tokenizers:
                vocab = tokenizer.get_vocab()
                g_prefix_count += sum(token.startswith("G") for token in vocab)
                u_prefix_count += sum(token.startswith("_") for token in vocab)

            # Determine dominant prefix type
            uses_g_prefix = g_prefix_count ¿= u_prefix_count
            print(f"  Detected prefix types: G=-g_prefix_count ,  _=-u_prefix_count ")
            print(f"  Using dominant prefix type: -'G' if uses_g_prefix else '_ ' ")

            # Process each tokenizer with normalized tokens
            for i, tokenizer in enumerate(self.tokenizers):
                try:
                    vocab = tokenizer.get_vocab()
                    normalized_tokens = set()

                    for token in vocab:
                        # Normalize tokens based on dominant prefix
                        if uses_g_prefix:
                            # Replace _ with G for consistency if this token uses _
                            if token.startswith("_"):
                                normalized = "G" + token[1:]
                            else:
                                normalized = token
                        else:
                            # Replace G with _ for consistency if this token uses G
                            if token.startswith("G"):
                                normalized = "_" + token[1:]
                            else:
                                normalized = token

```

```
                        normalized_tokens.add(normalized)

                    # Add normalized tokens to union vocabulary
                    self.union_vocab_set.update(normalized_tokens)

                except Exception as e:
                    print(f"  Error processing tokenizer -i : -e ")

            # Create sorted list and mappings
            self.union_vocab_list = sorted(list(self.union_vocab_set))
            self.union_vocab_token_to_idx = -
                token: idx for idx, token in enumerate(self.union_vocab_list)

            self.union_vocab_idx_to_token = -
                idx: token for token, idx in self.union_vocab_token_to_idx.items()


            # Store the dominant prefix type for later use
            self.uses_g_prefix = uses_g_prefix

            print(f"  Union vocabulary size: -len(self.union_vocab_list) ")
            if not self.union_vocab_list:
                print("[ERROR] Union vocabulary is empty!")
```

Listing 3: Whole create_vocab method

References

[1] Open LLM Leaderboard - a Hugging Face Space by open-llm-leaderboard. URL https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

[2] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language, November 2019. URL http://arxiv.org/abs/1911.11641. arXiv:1911.11641 [cs].

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. URL http://arxiv.org/abs/2005.14165. arXiv:2005.14165 [cs].

[4] Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi

Yang, Hailong Sun, and Philip S. Yu. Harnessing Multiple Large Language Models: A Survey on LLM Ensemble, May 2025. URL http://arxiv.org/abs/2502.18036. arXiv:2502.18036 [cs].

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, November 2021. URL http://arxiv.org/abs/2110.14168. arXiv:2110.14168 [cs].

[6] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding, January 2021. URL http://arxiv.org/abs/2009.03300. arXiv:2009.03300 [cs].

[7] Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. Ensemble Learning for Heterogeneous Large Language Models with Deep Parallel Collaboration, May 2024. URL http://arxiv.org/abs/2404.12715. arXiv:2404.12715 [cs].

[8] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension, May 2017. URL http://arxiv.org/abs/1705.03551. arXiv:1705.03551 [cs].

[9] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin,

Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

[10] Luis A. Ortega, Rafael Cabañas, and Andrés R. Masegosa. Diversity and Generalization in Neural Network Ensembles, February 2022. URL http://arxiv.org/abs/2110.13786. arXiv:2110.13786 [cs].

[11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

[12] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale, November 2019. URL http://arxiv.org/abs/1907.10641. arXiv:1907.10641 [cs].

[13] Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Zdeněk Kasner, Khalid, LSinev, Jeffrey Hsu, Anjor Kanekar, KonradSzafer, and AndyZwei. EleutherAI/lm-evaluation-harness: v0.4.3, July 2024. URL https://zenodo.org/records/12608602.

[14] Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. Hit the Sweet Spot! Span-Level Ensemble for Large Language Models, September 2024. URL http://arxiv.org/abs/2409.18583. arXiv:2409.18583 [cs].

[15] Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. Bridging the Gap between Different Vocabularies for LLM Ensemble, April 2024. URL http://arxiv.org/abs/2404.09492. arXiv:2404.09492 [cs].

[16] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Quick and (not so) Dirty: Unsupervised Selection of Justification Sentences for Multi-hop Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2578–2589, 2019. doi: 10.18653/v1/D19-1260. URL http://arxiv.org/abs/1911.07176. arXiv:1911.07176 [cs].

[17] Yao-Ching Yu, Chun-Chih Kuo, Ziqi Ye, Yu-Cheng Chang, and Yueh-Se Li. Breaking the Ceiling of the LLM Community by Treating Token Generation as a Classification for Ensembling, September 2024. URL http://arxiv.org/abs/2406.12585. arXiv:2406.12585 [cs].