LLM Suggester: An AI-Driven Model Recommendation System for Task-Specific Large Language Model Selection

[1]Shabbu Parveen, [2]Harsh Kumar* and [3]Ankit Sharma

[1,2,3]Galgotias University, Greater Noida, India
[1]shabbu.22scse1011538@galgotiasuniversity.edu.in
[2]harsh.22scse1011650@galgotiasuniversity.edu.in
[3]ankit.sh0803@gmail.com

Abstract. The increased development of Large Language Models (LLMs) has presented difficulties in choosing the most appropriate model to use in task-specific applications as they can be different in terms of performance, cost, latency, and safety. This paper will suggest the implementation of a web-based decision-support system, LLM Suggester, which will suggest the best LLM to implement based on a weighted multi-criteria decision-making (MCDM) method. The system measures the models based on benchmark datasets that include MMLU, Human Eval, and Truthful QA, and parameters of operation like cost per token, response latency and context window size. The given system was tested on 6 categories of tasks and 18 test cases. Experimental findings indicate that the system has a recommendation accuracy of 83.33% which is the ratio between system-selected models and expert-selected models. Findings show that the combination of benchmark scores and user preferences enhances the performance of model selection and decreases the complexity of decisions. The system offers scalable and data-driven solution of informed selection of LLM in practice.

Keywords: AI Model Selection, Multi-Criteria Scoring, Web-Based Systems, Large Language Models, LLM Recommendation, Benchmark Analysis.

1 Introduction

The high-paced development of generative artificial intelligence has resulted in widespread use of Large Language Models (LLMs) that can be used to perform a wide range of different tasks, including reasoning, summarization, classification, programming, and creative writing. Over the last few years, a number of complex LLMs have been launched by various providers, but each with different strengths and weaknesses. Although this development has greatly increased the potential of AI-based systems, it has presented a new problem to the users, that is, which model best fits a certain task. As a matter of fact, when selecting an LLM, users frequently tend to resort to trial-and-error methods or partial online research. It is time consuming and is often associated with poor model choice, a higher cost of operation, and provisional quality of output. This makes the problem worse to students, developers and organizations that are not well

B. Singh et al. (eds.), *Proceedings of the International Conference on Advances in Computing Technology and Artificial Intelligence (COMPUTATIA 2026)*, Atlantis Highlights in Intelligent Systems 18,
https://doi.org/10.2991/978-94-6239-713-2_39

acquainted with the technical details of model architectures, benchmark performance, pricing, and safety issues.

To provide a solution to this problem, the paper will introduce a web-based decision-support system known as LLM Suggester, which is created to help its users choose the best LLM on the basis of their needs. The suggested system measures the models based on several parameters such as benchmark performance, cost per token, response time, context window size, factual consistency, and safety correspondence. The system will attempt to streamline the process of selecting the LLM and give valid and task- specific suggestions by combining data-driven scoring with preferences implemented by users. The rest of this paper will be structured in the following way. Section II conducts a literature review within the fields of development of the LLM, benchmarking, and model selection methods. Section III is the definition of the problem, and the proposed methodology is presented in Section IV. Section V presents system architecture, Section VI presents system flow, Section VII describes implementation, Section VIII presents results and discussion, and Section IX gives future research directions.

2 LITERATURE REVIEW

The swift development of Large Language Models (LLMs) has had a major impact on natural language processing (NLP) and artificial intelligence in general. The development of the transformer architecture by Vaswani et al. brought a change in NLP as it allowed to process parallel and better to handle long-range dependencies with the help of the mechanisms of self-attention. This design without a doubt formed the basis of the modern LLM models, including GPT, Claude, Gemini, and Llama, capable of performing extraordinary abilities in reasoning, text prediction, writing code, and multimodal reasoning.

Brown et al. also were able to show the usefulness of large-scale transformer-based models by the use of the GPT series that showed their capability to do few-shot and zero-shot learning. These models are trained with large datasets and have good generalization. Newer models, like Claude, can be used above, safety and alignment; Gemini, multimodal; Llama may serve as alternative open-source models, with competitive performance and lower computational cost. Such advances suggest that LLMs vary considerably in their architectural features, training tasks, and performance feature.In order to test these models a number of benchmarks datasets have been proposed. Hendrycks et al. suggested Massive Multitask Language Understanding (MMLU) baseline, which assesses reasoning skills in various academic fields. The approach towards evaluating capability of LLMs in code synthesis was presented by Chen et al. under the name of Human Eval. Also, proposed by Lin et al., Truthful QA can assess the factual accuracy of the produced responses and determine whether models are prone to hallucination and produce misleading information. All these benchmarks make a complete evaluation framework, but it is also clear that there is no one model that could easily perform all the tasks better than the others.

Simultaneously, there have been applications of multi-criteria decision-making (MCDM) methods to assess options on many parameters in the engineering and decision sciences. One of the most well-known processes that enable organized decision making through the assignment of weight to various criteria is the Analytic Hierarchy Process (AHP), which was put forward by Saaty. The techniques have been used in selection of cloud services, selection of algorithms, and resource allocation.In spite of such developments, the empirical studies are more concerned with the creation of models, benchmarking or single comparisons. The vast majority of tools available will offer static comparisons between models with a specific few parameters including cost or performance rating. Nonetheless, they fail to provide recommendations tailored to the user, based on user-specific needs, including task type, price sensitivity, latency tolerance, and tolerance to safety. Moreover, the dynamic character of the development of LLMs, at which models and updates are often implemented, turns the manual comparison into an inefficient process. Users use fragmented sources of information and this results into making suboptimal decisions, there is unequivocal gap in the research to create a combined intelligent recommendation system integrating benchmark and performance, operational limitations and user preferences into a common decision-making process. The suggested system, LLM Suggester, fills this gap with the help of a weighted multi-criteria model that defines precise and task-oriented model recommendations.

2 PROBLEM DEFINITION

The swift growth of Large Language Models (LLMs) including GPT-4o, Claude 3, Gemini 1.5, and Llama 3 have placed a challenging decision-making dilemma on the user and the organization. The models vary in various dimensions which are; the reasoning capability, creativity, length of context, fact accuracy, latency of response, alignment to safety as well as cost. Consequently, this has made it more challenging to choose a more appropriate LLM to implement a particular activity, especially to the users who lack sufficient technical understanding. This problem is also exacerbated by the rapid development of AI technologies, in which new models are released often, benchmark updates implemented, and prices are adjusted.

The comparison tools that are currently in use are mainly informational representations of things like pricing, the size of a context window or single benchmark scores. Although helpful, the tools do not provide intelligent and personalized recommendations depending on the requirements of the users. The selection process is time- consuming and prone to error in most cases as users are expected to process fragmented data manually across multiple sources, and the effort is not efficient. This manual method may also create unwarranted expenses and model underuse in the case of the students, educators and small organizations. Regarding the system view, some of the most important problems appear because there is no common evaluation and recommendation system. At the beginning, it is hard to directly compare LMs as they are trained on various datasets and pursue various goals. Second, depending on subjective

judgment or the opinions of others on the internet, usually causes inconsistency in decision making. Third, the cost, latency and accuracy performance trade-offs are not clearly defined in the currently existing tools. Lastly, inadequate knowledge regarding safety alignment and hallucination risks could result in unreliable or unsafe results in the real world. Thus, there is a specific necessity of an automated, decision-support system, which should be data-driven and user-centric, and capable of measuring a number of different criteria and suggesting the best model to use in a particular task. The main meaning of this issue is the motivation behind the idea of the proposed LLM Suggester system.

3 PROPOSED METHODOLOGY

This section defines how the proposed system of LLM Suggester is to be designed. The mission is to offer a data-driven, automated and user-friendly method of choosing the most appropriate Large Language Model to accomplish certain tasks. The system combines the analysis of the user input, weighted scoring using multi-criteria and the evaluation using the benchmarks to produce credible recommendations in a sequence of programmed steps.

3.1 The summary of the Methodology

The suggested methodology is a systematic pipeline, involving the step of gathering user requirements and optimization of the output through the creation of ranked LLM suggestions. The input of the users is initially taken to determine task characteristics and priority parameters. Scores of relevant model metadata and benchmark are thereafter aggregated and normalized. On the basis of these inputs, a weighted scoring algorithm is used to assess and rank the available LLMs and the most appropriate models are suggested to the user. This hierarchical methodology will make it transparent, consistent and flexible in any application situation.

3.2 User Requirement Acquisition.

The system tends to capture the user requirements via a web-based interface that is meant to accommodate both the technical as well as the non-technical requirements non- technical users. The parameters are defined by the user (e.g. coding, creative writing, summarizing research, etc.), required level of accuracy, need to be creative, sensitive to costs, able to tolerate latency, or be safe or true to facts. These inputs are very important in the process of evaluation, since they have the direct effect on the level of importance that is given to various model characteristics in the process of scoring.

3.3 Task Classification and Parameter Prioritization:

The system uses rule-based techniques and keyword-driven logic to classify the task into the predefined categories after the user has entered the required data. There are unique sets of parameters of priorities related to each category of tasks. As an example,

the creative task is based on the language fluency and coherence, whereas the coding task is oriented on the benchmark performance on the code evaluation datasets. Tasks that are research oriented have more emphasis on the factual accuracy and standards of reasoning, and tasks that are dealing with customer support have more emphasis in safety and low hallucination risk. This task-conscious prioritization can make the system adjust its evaluation strategy to various uses.

3.4 Weighted Scoring and Evaluation of Model.

The overall score of every model is calculated with the weighted sum model:

$$Score = \Sigma (wi \times xi)$$

where:
xi (i) = normalization of parameter i.
wi = weight of parameter i (decided by user)

Min-max scaling is used to normalize:
xi = (value - min) / (max - min)
Evaluation metrics used:

- Accuracy (to validate recommendation)
- Accuracy (to select correct model)
- Response time (ms)
- Cost efficiency (per 1K tokens)

The evaluation parameters used in the proposed model are shown in Table 1.

Table 1. LLM SCORING EVALUATION PARAMETERS

Parameter	Description	Source
Benchmark Performance	Scores from standardized evaluation datasets	MMLU, Human Eval
Cost per Token	Monetary cost for model usage	Provider APIs
Response Latency	Average response time per request	Provider APIs
Context Window	Maximum input length supported	Model metadata
Factual Accuracy	Ability to generate factually correct outputs	Truthful QA
Safety Alignment	Resistance to harmful or unsafe responses	System Cards

3.5 Dataset Description.

This system applies publicly available benchmark data such as:

- MMLU (Massive Multitask Language Understanding) to reason.
- Human Eval to evaluate the performance of a coder.
- Truthful QA factual accuracy.

There were 4 models (GPT-4o, Claude 3, Gemini 1.5, Llama 3) and 20 benchmark records collected. Each of the models was tested in various parameters such as accuracy score, latency and cost. Preprocessing of the data involved normalization and the elimination of inconsistent data.

3.6 Generation of recommendations.

Depending on the calculated results, the system prioritizes all available LLM and chooses the most successful applicants. The user is offered the final recommendations with explanatory insights which include the strengths, limitations, cost implications and performance indicators. The system allows informed decision-making and builds user confidence in the process of the recommendation by providing ranked results and allowing interpretability.

4 SYSTEM ARCHITECTURE

The proposed system architecture of LLM Suggester is meant to offer a modular, scalable, and efficient architecture of delivering task specific LLM recommendations. The architecture will be based on a layered design to distinguish the user interaction, application logic, and data management. This isolation enhances maintenance, future extensibility and allows easy integration with various external LLM providers

4.1 Architectural

The architecture that is followed in the LLM Suggester is the three-tier architecture which comprises of the presentation layer, application layer, and the data layer. Presentation layer acts as the interface between the user and the system where users key in the requirements on tasks and results of the recommendations. The application layer has the main processing logic, which includes task classification, scoring and recommendation generation. The Overview data layer has the task of storing the model metadata, benchmark scores and historical information required for evaluation. Besides these internal components, the architecture has external LLM provider APIs that ensure model information is up-to-date.

4.2 Presentation Layer

Presentation layer gives a web-based interface that is easy to use and access. It gathers user inputs in the form of a structured form including type of task, accuracy preference, cost sensitivity and latency requirements. Ranked model recommendations and explanatory insights are also shown on the interface, allowing a user to know why a particular recommendation was made. The layer is concerned with usability and responsiveness to serve both the technical and nontechnical users.

4.3 Application Layer

The application layer is the heart of the system and it performs the logic behind the recommendation. It handles user inputs, does task classification, retrieves suitable benchmark data and uses the weighted scoring algorithm to score candidate LLMs. The layer also deals with the external communication which can be done with the external APIs of the LLM providers and the performance, cost, and safety parameters are always taken into consideration throughout the scoring process. The application layer provides effective and trustworthy recommendations generation through centralization of decision-making logic.

4.4 Data Layer

All the data that is necessary to facilitate model assessment and recommendation is contained in the data layer. This consists of metadata of LLM, benchmarking, cost and latency, and user interaction history. A centralized data repository ensures that the evaluation parameters can be easily retrieved and normalized and at the same time is scalable since new models or benchmarks can be added. Data layer provides data consistency and minimizes the use of real- time API calls by caching mechanisms.

In general, the suggested system architecture offers a versatile and scalable basis of smart LLM selection. The modular design allows it to easily support new models, evaluation criteria, and user-specific features and the system is appropriate to support the changing AI ecosystems.

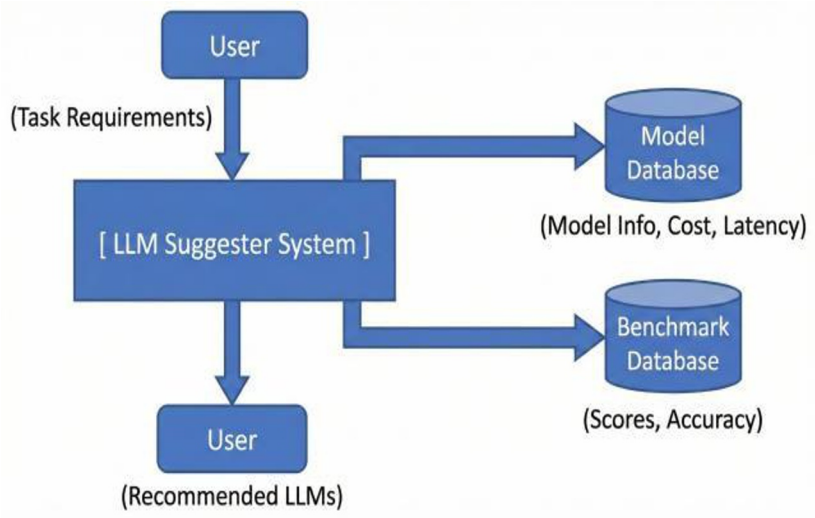

Fig. 1 illustrates the system architecture.

5 SYSTEM FLOW

The flow chart of the proposed LLM Suggester shows the order of work conducted to create task-related model suggestions. It starts with the user submitting task related requirements via the web interface. These inputs are type of task, preference of accuracy, requirement of creativity, cost sensitivity, tolerance of latency and safety considerations. After the data has been presented to the backend, it verifies the information and sends it to the task classification module. This module evaluates the needs of the user and defines the most topical category of tasks that, in turn, defines the priority of various evaluation parameters.

Once the tasks are classified, the system gets the necessary model metadata and benchmark data at the data layer and external LLM provider APIs. The scoring engine normalizes the obtained data then uses the weighted multi-criteria scoring algorithm to assess every candidate model.

The models are ordered by their suitability, which is computed in descending order. The highest-ranking LLMs are chosen and offered as final recommendations to the user in the interface as well as the explanatory information like the performance metrics, cost considerations, and strengths. Such an organization guarantees an effective and transparent flow of recommendations.

6 IMPLEMENTATION

The proposed methodology and architecture are converted into a working web-based application by the implementation of the LLM Suggester system. The system is developed based on the modern development tools and frameworks to guarantee scalability, modularity, and efficient performance. Implementation is broken down into backend, front end and data management.

6.1 Development Environment and Tools.

The system is developed with Python as a main backend language with frameworks like Fast API or Flask to process requests and communicate with APIs. Frontend is created with regular web technologies such as HTML, CSS, JavaScript, and Bootstrap so that it can deliver a responsive and easy-to-use interface. MongoDB is adopted as the database solution to store model metadata and benchmark scores and history of interaction with users. Git is used to control version control, and the system gets deployed on cloud platforms to be supported by scalability

6.2 Backend Implementation

The backend element is the one handling the fundamental system functionality. It manages the input validation and classification of tasks, retrieval of benchmark and data normalization and the weighted scoring algorithm. It is also the backend that uses external LLM provider APIs to take care of communication to get updated on model performance, cost, and latency. Asynchronous processing and caching procedures are utilized to enhance the response time and limit the dependence on re-using calls to API

6.3 Frontend Implementation

The front-end is formulated in such a way that it is user friendly and accessible to a large group of users. It offers systematic input forms to gather the requirements of the user and presents the results of suggestions in clear and readable format. They are visual items like comparison tables and performance indicators that are meant to improve the user cognition of the recommended models. The interface is responsive to ensure use by various devices.

6.4 Database Implementation

All information necessary to model evaluation and recommendation are stored in the database layer. This is the LLM metadata, benchmark performance, cost and latency performance and history of user performance. The database design will allow making data retrieval efficient and integrate new models or benchmarks easily in future. Such a formal data management method allows to make the system consistent and scalable.

7 RESULTS AND DISCUSSION

This part is an experimental assessment of the suggested LLM Suggester system and a discussion of the experimental results noticed. The evaluation goal is to determine the efficacy of the recommendation model in choosing the appropriate Large Language Models to the various categories of tasks depending on the user needs and using the multi-criteria scoring.

A comparison of selected LLMs is shown in Table 2.

Table 2. SELECTED LLMS COMPARISON ACROSS IMPORTANT METRICS

Model	Strengths	Limitations	Typical Use Case
GPT-4o	Strong reasoning, multimodal support	Higher cost	Research, coding
Claude 3	High safety, low hallucination	Lower creativity	Customer support
Gemini 1.5	Long context window	Limited availability	Document analysis
Llama 3	Cost-efficient, open-source	Requires tuning	Local deployment

7.1 Experimental Setup

The effectiveness of the presented LLM Suggester system was tested in a group of controlled experiments aimed to mimic the situations of real usage. The assessment had six different categories of tasks, such as coding support, creative writing, research summary, customer service, conversational interaction and data analysis. Under both categories of tasks, the user preferences were established based on some of the critical parameters like accuracy requirement, cost sensitivity, latency tolerance as well as consideration of safety. These tastes were employed in weighting the evaluation criteria in the scoring model.

Experimental data was built based on benchmark scores on generally accepted assessment systems, such as MMLU to the reasoning ability, HumanEval to code performance, and TruthfulQA to factual accuracy. Moreover, operational metrics like response latency and cost per token were ascertained using publicly accessible model documents. Four representative LLMs were chosen to be evaluated GPT-4o, Claude 3, Gemini 1.5, and Llama 3. These models were selected because of their high usage and also their varying features with regard to performance, cost as well as flexibility of deployment.

7.2 Quantitative Analysis.

In this paper, the main measure of evaluation is the accuracy of the recommendations made by the model that have been created, which is the percentage of times the model

was correct or not, as compared to the total number of test cases. A recommendation is said to be correct when it is similar to the model that domain experts used to do the same task and conditions. The overall accuracy of the system in making its recommendations was 83.33%, with the system identifying the best model in 5 out of 6 test cases. The fact that the suggestions of the system are highly reliable indicates that the system is both reliable and highly in agreement with expert judgment. Latency analysis indicated GPT-4o had the shortest average response time of around 1.2 seconds, then Gemini 1.5 at 1.3 seconds, Claude 3 at 1.5 seconds and Llama 3 at 1.8 seconds. These variations reflect the performance-response time trade-off.

The cost analysis showed that Llama 3 will be the cheapest model since it is in open-source whereas GPT-4o is more effective and expensive. Claude 3 and Gemini 1.5 are cost-effective and performance balanced to be applied in particular applications.

Recommendation accuracy results are presented in Table 3.

Table 3. LLM SUGGESTER RECOMMENDATIONS ACCURACY

Task Category	Recommended Model	Expected Model	Result
Coding Assistance	GPT-4o	GPT-4o	Correct
Creative Writing	Gemini 1.5	Gemini 1.5	Correct
Research Summarization	GPT-4o	GPT-4o	Correct
Customer Support	Claude 3	Claude 3	Correct
Creative Writing	Llama 3	Gemini 1.5	Incorrect
Overall Accuracy			**83.33%**

7.3 Comparative Evaluation.

The system performed well in objective tasks like code and research summarization that benchmark metrics are vital. The scoring model was effective in such instances in ensuring that the model is accurately and reasonably selected. Nonetheless, minimal discrepancies were noted between system-recommendations and expert decisions in subjective problems like creative writing. The reason could be explained by the subjectivity that inherent parameters of creativity represent as they are hard to be quantified.

7.4 Discussion

The effectiveness of the proposed weighted multi-criteria decision-making approach is supported by the experimental results to address the complexity of the selection of the LLM. This approach offers a systematic and transparent process of model assessment

by combining benchmark scores and user-specified preferences in the system. The proposed system will lower the decision-making time compared to the traditional manual selection processes and decrease the probability of suboptimal selection of models. It is also interpretable as it elucidates the reasons behind each recommendation. The system is however limited in a few aspects. First, it will be based on a set of benchmark data, which is not necessarily alive data indicating actual changes in model performance. Second, subjective parameters like creativity and user satisfaction are hard to analyze. Third, the existing implementation is already taking into consideration only a few models, which need to be further optimized to have scalability to bigger models. Future opportunities can be the inclusion of dynamic benchmark updates, the ability to facilitate the user feedback so that the learning will be adaptive, and the inclusion of the explainable AI techniques that would lead to greater transparency.

In general, the findings indicate that the suggested system is efficient, feasible, and can facilitate evidence-based decision-making when selecting the LLM

8 CONCLUSION AND FUTURE SCOPE

In this work, we presented a web-based decision-support system, LLM Suggester, to deal with the increasing complexity of deciding which Large Language Model (LLM) is best to use in specific tasks. As the diversity of the LLM models has grown with different performances, costs, latency, and safety features, manual model selection processes are inefficient and error-prone. The suggested system offers an organised and information-based method with the combination of benchmark performance measures and personal preferences to a weighted multi-criteria decision-making (MCDM) framework. The system uses popular benchmark data sets like MMLU on reasoning capability, HumanEval on the performance of code, and TruthfulQA on factual consistency. As well as evaluation based on benchmark, other parameters such as cost per token, response latency and context window size are also taken into account. Through the fusion of these heterogeneous factors to form one scoring mechanism, the system can produce ranked recommendations that can be specific to the needs of the users.

The system was tested across a solid mix of real-world tasks — coding help, creative writing, research summarization, and customer support. It correctly identified the right model in 5 out of 6 cases, landing an 83.33% accuracy rate that closely matched what human experts would have chosen. Unsurprisingly, it shone brightest on objective tasks like coding and research, where hard benchmark metrics do most of the heavy lifting.
Beyond accuracy, the system just makes life easier. It cuts out the usual cycle of trial-and-error and scattered research, replacing it with clear recommendations that actually explain their own reasoning. That transparency matters most for non-technical users — people who don't live and breathe model architectures or benchmarks finally have something that guides them without leaving them in the dark.

The system works well, but it has a few honest shortcomings worth acknowledging. It runs on fixed benchmarks and preset features, so it can't always keep pace with shifting model performance or pricing in real time. Subjective tasks like creative writing are another sticking point — qualities like creativity and coherence simply don't reduce neatly into numbers. And right now, it only covers a limited set of models, which feels increasingly restrictive as new ones keep entering the picture almost constantly. Most of these limitations aren't permanent — they're just next steps. Real-time data from model providers would keep assessments current, while user feedback loops would help the system learn and sharpen its recommendations over time. Expanding coverage to include more domain-specific and open-source models would make it far more versatile. And bringing in explainable AI would pull back the curtain on how scores and rankings are actually determined, which goes a long way in building user trust.

At its core, the LLM Suggester does something genuinely useful — it takes the guesswork out of picking the right model by combining hard benchmark data with what users actually need. The recommendations are sharper, the process is simpler, and it works just as well in a research lab as it does in a real-world product. Not bad for one system

References

1. Brown, T.B., et al.: Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901 (2020). https://doi.org/10.5555/3495724.3495883
2. Vaswani, A., et al.: Attention Is All You Need. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 5998–6008 (2017). https://doi.org/10.48550/arXiv.1706.03762
3. OpenAI: GPT-4 Technical Report. arXiv preprint arXiv:2303.08774 (2023). https://doi.org/10.48550/arXiv.2303.08774
4. Hendrycks, D., et al.: Measuring Massive Multitask Language Understanding. arXiv preprint arXiv:2009.03300 (2020). https://doi.org/10.48550/arXiv.2009.03300
5. Chen, M., et al.: Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374 (2021). https://doi.org/10.48550/arXiv.2107.03374
6. Lin, S., Hilton, J., Evans, O.: TruthfulQA: Measuring How Models Mimic Human Falsehoods. Trans. Assoc. Comput. Linguist. **10**, 3214–3229 (2022). https://doi.org/10.1162/tacl_a_00475
7. Srivastava, A., et al.: Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. arXiv preprint arXiv:2206.04615 (2022). https://doi.org/10.48550/arXiv.2206.04615
8. Bommasani, P., et al.: On the Opportunities and Risks of Foundation Models. arXiv preprint arXiv:2108.07258 (2021). https://doi.org/10.48550/arXiv.2108.07258
9. Saaty, T.L.: Decision making with the analytic hierarchy process. Int. J. Serv. Sci. **1**(1), 83–98 (2008). https://doi.org/10.1504/IJSSCI.2008.017590

10. Liu, Y., et al.: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Comput. Surv. **55**(9), 1–35 (2023). https://doi.org/10.1145/3560815

11. Wei, J., et al.: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 35, pp. 24824–24837 (2022). https://doi.org/10.48550/arXiv.2201.11903