Machine Learning Delegation Based Ensemble Method For Large Language Models

Ojaswi Bhimineni
University of Michigan
ojaswib@umich.edu

Clark Kaminsky
University of Michigan
clarkkam@umich.edu

Joshua Symonds
University of Michigan
jsymonds@umich.edu

Abstract

In the era of consumer-oriented Large Language Models (LLMs) like ChatGPT and Google Bard, end users often treat these ChatBots as all-knowing black boxes and expect accurate responses to any question without realizing the limitations of the underlying LLM. Numerous specialized LLMs excel in specific domains and outperform these generalized models, yet remain underutilized because users are often unaware of their existence or their superior performance within that domain.

To address this challenge, we present a delegation-based LLM ensemble approach. This method enables users to engage with a unified interface for their conversational queries while exploiting the domain specialization of an ensemble of LLMs. When presented with a user prompt, a machine learning delegator identifies the most suitable LLM from its ensemble based on the prompt's question style and informational domain and the delegator then leverages the best model to generate a response. This ensemble approach empowers users to access the collective expertise of a wide range of specialized LLMs without requiring them to manually identify and utilize the best model for their task.

This work aims to bridge the gap between user expectations and LLM capabilities, enhancing the overall effectiveness and user experience when interfacing with large language models.

1 Introduction

LLMs' sizes have been increasing exponentially over the past decades, which has led to more and more powerful and capable models. As seen in Fig 1, the number of parameters in state-of-the-art LLMs has been increasing by a factor of nearly 10 each year for the past five years, allowing them to better handle many specific use cases and be applicable to wider market domains.

With their meteoric rise in both power and public accessibility, these large models have been adapted and pre-trained to solve specific types of problems that are aligned to the models' structures. The diversity and capability of modern LLMs has led to the prominent question of this paper: how can we combine and utilize various LLMs' breadth of knowledge under a single interface?

To address this challenge, maximizing the utility of diverse LLMs, we propose a delegation-based LLM ensemble approach. This method facilitates user interaction through a unified interface for queries while leveraging the domain specialization of an ensemble of LLMs. When confronted with a user prompt, a machine learning delegator identifies the most suitable LLM from its ensemble based on the question style and informational domain, then utilizes the selected model to generate a response, allowing users to access the collective expertise of a wide range of specialized LLMs without having to understand or predict themselves which LLM is best for their circumstance.

This work not only addresses the widening gap between user expectations and LLM capabilities but also delves into the challenge of combining and utilizing the growing breadth of knowledge embedded within various LLMs. By presenting a delegation-based LLM ensemble approach, we aim to enhance the overall effectiveness and user experience in interfacing with these large language models. Through this research, we seek to provide a solution that aligns the widespread advancements in LLM capabilities with user expectations, offering a more seamless and informed interaction environment.

2 Related Work

Based on the training objectives and the datasets that open-source pretrained LLMs were trained on, their performance and functionality varies based

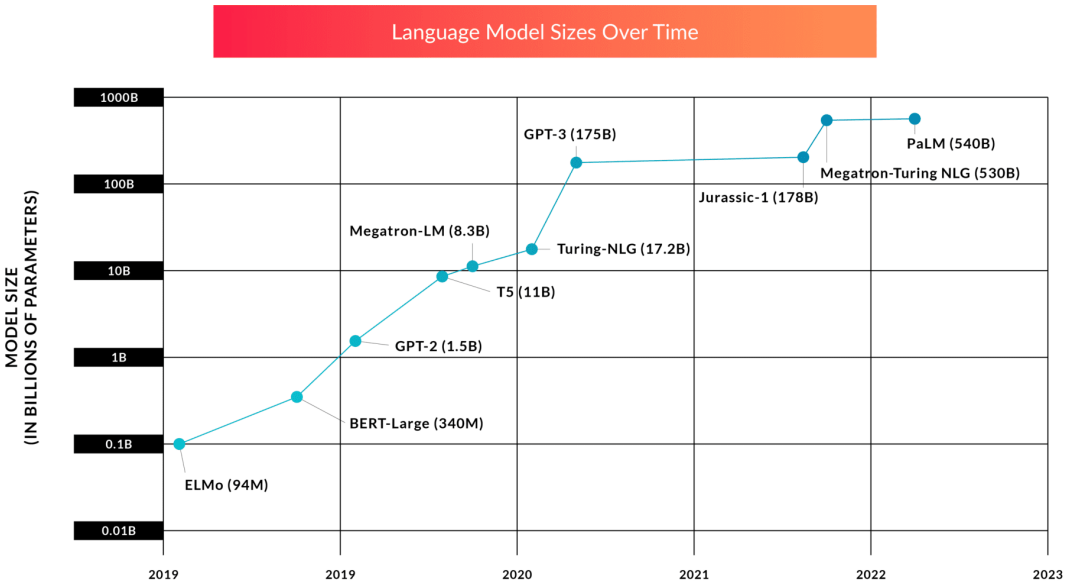

Figure 1: LLM Sizes Over Time

on question types and formats. Some LLMs excel at specific tasks while failing to correctly address others. The comprehensive overview from Naveed et al. [8] shows the vastly different tasks that different LLMs employ, including Question Answering, Classification, Machine Translation, Reading Comprehension, Mathematical Reasoning, Coding, and Memorization. Most of the models are only able to achieve success in a few of the categories consistently.

For example, the GPT family of LLMs are very successful at answering essay based and text-generation based questions, while showing difficulty of answering multiple choice questions about code.[11] [9]

Similarly, many language models struggle with basic arithmetic operations, especially with higher numbers of digits, but the paper by Yang et al. [12] introduces a model called MathGLM that can consistently and accurately answer both arithmetic and mathematic-based word problems.

Since this is a relatively new idea, there are few papers on the subject of choosing an optimal LLM for a given question. Jiang et al. [4] have introduced a novel way to address this problem and capitalize on the different strengths of various LLMs, by creating an ensemble learning method to combine the outputs of separate models. As seen in Fig 2, this paper finds that the optimal choice for different example questions rarely favors one model, with more than 5 of the 11 models they incorporated being the best choice more than 5% of the time. The article additionally demonstrates that their "LLM-Blender" achieves a significant improvement over the next best model. Another important novelty from this paper is a new dataset they call MixInstruct, which is a set of instruction examples used to train and test ensemble models.

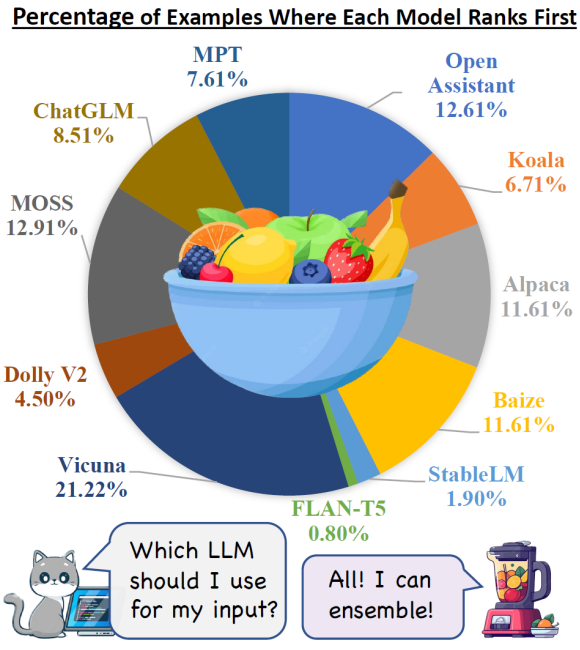

Figure 2: Jiang et al. [4] LLM Blender

3 Methodology

This section outlines our approach to training a machine learning model for question domain identification. We combine diverse questions from the five aforementioned domains by preprocessing data, utilize a Recursive Neural Network, and evaluate the model based on accuracy. Our methodology also includes discussion of model selection in 3.1 and the development of an interface for user interaction.

The diagram of our model can be found in Fig 3.

1. **Data Collection** - We were unsatisfied with any question category labelling datasets we were able to find publicly available so we code to develop out own dataset to train our machine learning delegator on. We chose to train our machine learning delegator to distinguish between five question style / domains:

   (a) Conversational text (CONV)
   (b) Logical or Scientific reasoning (SCIENCE)
   (c) Programming questions (CODE),
   (d) Mathematical questions (MATH)
   (e) Legal / Law questions (LAW)

   We chose these categories since we felt they were distinct enough that a human can easily distinguish which category a question would belong to, but similar enough that a machine learning model would struggle to distinguish them without proper training. Specifically, we anticipated that science and math, conversational and science, and math and programming, and conversational and law would be difficult pairs for the model to distinguish. We also chose these five categories since we expect that models fine tuned in responding to prompts in each of these domains would dramatically outperform a model which is not specialized within their domain, which would exemplify the improvements that our design offers over using a single general model. The specific datasets we used for each domain can be found in Table 1.

2. **Data Preprocessing** - Since our different question categories are dramatically different, the questions are often asked in different formats or with different contexts. To create a uniform dataset, we isolate the question or prompt from the surrounding context in order for our model to make its decisions based only on the prompt (as opposed to recognizing the format of the context). An exception to this however, is the CODE questions - we felt that without the code as context, the questions could not make sense so we chose to keep the code as part of the prompt. After processing our data into only the necessary prompt, we tag each question with the question type and shuffle them into our train, validation, and test sets which had 15000, 4123, and 1250 rows each. Since the size of our datasets differed dramatically (AQuARAT had about 100,000 rows and ARC has about 3,000 rows for training data), we stratified our data to have no more than 3,000 rows from any category for training, 1,000 for validation, and 250 for testing.

3. **Machine Learning Model Training** - We then trained a machine learning model to identify which style of question a given prompt is. This model is trained on the dataset we gath-

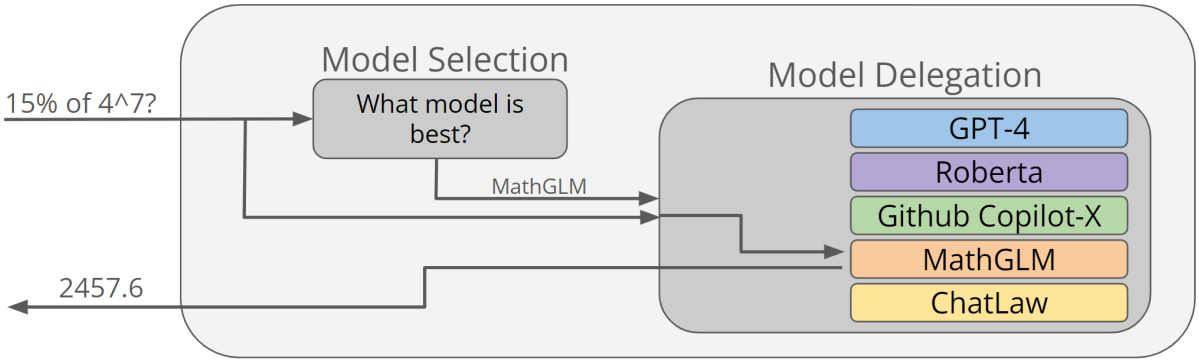

Figure 3: Model Diagram

Table 1: Question Type Data sets

Question Type	Data set
CONV	Conversational Question Answering [10]
SCIENCE	AI2 Reasoning Challenge (Easy and Challenge) [1]
CODE	CodeQA [6]
MATH	Algebraic Question Answering With Rationales (AQuA-Rat) [5]
LAW	CaseHOLD [13]

ered and generated, and its results are used to select which large language model in the ensemble is utilized to best respond to the prompt. Much more detail about the model training process can be found in Section 3.1.

4. **Large Language Model Selection** - After identifying which question type a given prompt is, we map the question type to the model we chose to respond to those questions. We heuristically chose models which seem to be most capable to respond to our five different question types, the specific models we chose can be found in Table 2

Table 2: Ensemble Large Language Models

Question Type	Model
CONV	GPT-4 [9]
SCIENCE	Roberta [7]
CODE	Github-Copilot-x [3]
MATH	MathGLM [12]
LAW	ChatLaw [2]

5. **Integration and User Interface** - Create an integrated system that routes questions to the appropriate model based on categorization results. Design a user-friendly interface for users to input questions and receive responses seamlessly. This interface can be seen in Fig 4.

3.1 Model Development and Training

We chose to use a LSTM (Long-Short-Term-Memory) Recursive Neural Network for our question type identification machine learning model due it its ability to recognize sequential characteristics of language in addition to word correlations. For our evaluation metric we chose to evaluate with respect to accuracy since it best represents our goal for the model

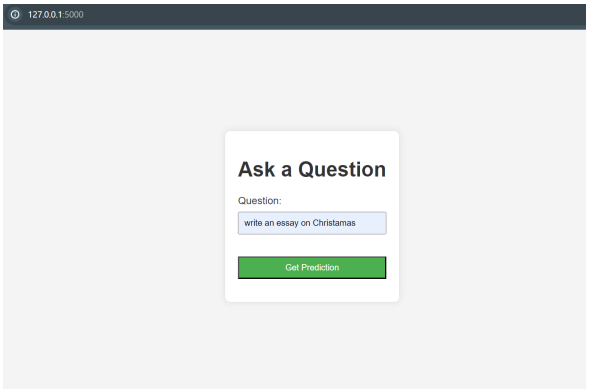

Figure 4: User Interface for Final Product

- to categorize correctly as often as possible. We use GloVe word embeddings to encode our prompts computationally. In order to improve our likelihood of selecting the model that best generalizes to our test data, we saved the model from the epoch which performs best on our validation dataset; for every epoch we record the model's accuracy on the training dataset as well as the validation dataset and the validation dataset partitioned by question label.

Our initial attempts at training our model seemed to both overfit to the training data (training accuracy was much higher than validation) and failed to extract meaningful features from training data (sporadic accuracy between training epochs showed a lack of convergence based on feature meanings). Despite these shortcomings, this model achieved about 70% testing accuracy. As a result of our first model training attempts suffering from both of these, we noted that more intentional approaches to hyperparameter tuning would be required than just picking which ones seem to improve performance. We split into two different approaches for rectifying

these issues - training a small regularized model which would avoid overfitting but may risk under fitting, and a much larger model which we expect to more accurately extract meaningful features from the training data at the risk of overfitting. Although our first method behaved as expected, performing decently with no overfitting - we incorporated the regularization strategies we learning through training the first approach to allow for the second approach to surpass our expectations and yield a very satisfying result which performs very well on the validation and test data with very little overfitting and it seems to extract high level features from the data as well.

3.1.1 Method 1: Small model

This attempt at training the model is most concerned with mitigating overfitting of the model. We chose to train a relatively small RNN with regularization and a high learning rate in the hopes that the model will quickly converge on the high-level features of the training data without overfitting too much. This model was able to overcome the overfitting, with validation and test accuracy that often surpassed the training accuracy. Although this model achieved very good performance (91% testing accuracy), it is apparent that the model was not effectively converging to an optimal set of weights. Between each epoch, the accuracy changes dramatically for the validation and training sets. Additionally, the accuracy - when split by question type is concerning unpredictable with some labels being categorized with 0% accuracy for an epoch on several occasions. Since our validation accuracy peaked on the third epoch, we recorded a model which performed decently on all of the question labels; as expected, it most struggled with distinguishing MATH and SCIENCE questions. The confusion matrix for our this method's test performance can be found in Fig 5, the accuracy while training can be found in Fig 6, and label-specific accuracy can be found in Fig 7. The specific hyperparameters for this model can be found in Table 3.

Table 3: Method 1 Model Hyperparameters

Hyperparameter	Value
Epochs	10
Learning Rate	0.05
Embedded Dimension	24
Hidden Dimension	16
Optimizer	SGD
Loss	CrossEntropy
Weight Decay	0.001
Momentum	0.9
LR Decay	N/A

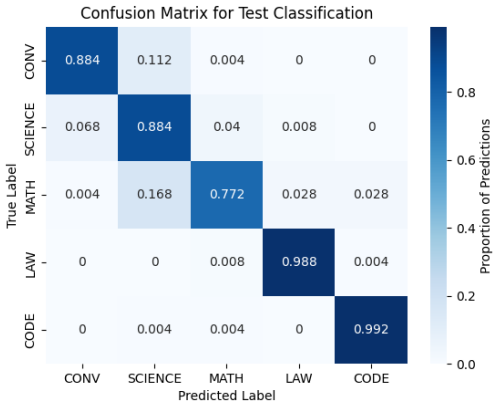

Figure 5: Method 1: Confusion Matrix

3.1.2 Method 2: Large model

Conversely to method 1, this method attempts to ensure that our model is large enough to effectively model the features in our data. With this method, we hope to see more consistent accuracy between epochs in general and for specific labels throughout training.
This method was a great success, the increased model sized allowed for us to perform much better and reliably between epochs. Further, with regularization techniques such as weight decay, momentum, and using a learning rate scheduler we were able to both perform very well and refrain from overfitting on training data. Further, we trained this model with a smaller learning rate and for several more epochs to help it converge to an optimal solution. The specific details of the model training can be found in Table 4. This model achieved 97.92% test accuracy which significantly surpassed our initial benchmark of 70% from our project proposal for a successful model.

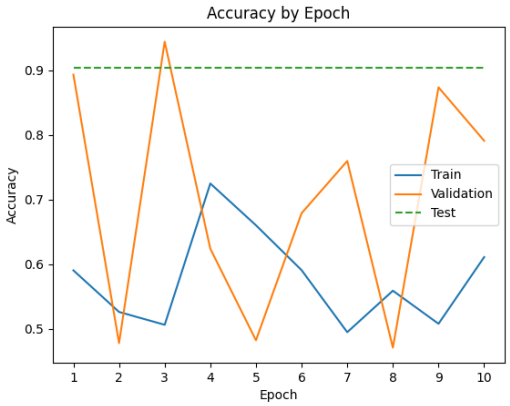

Figure 6: Method 1: Training Accuracy

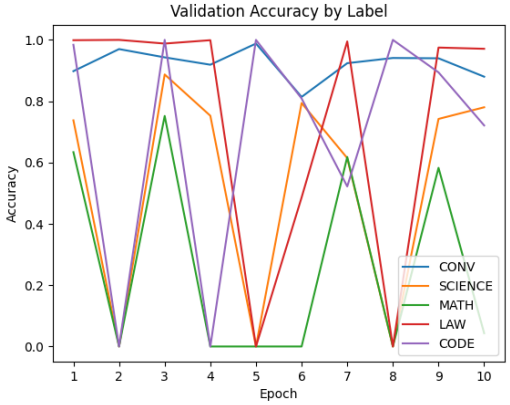

Figure 7: Method 1: Accuracy by Question Type

This model performed as expected with its misclassifications, with the vast majority of them being between CONV and SCIENCE and SCIENCE and MATH - the two pairs of labels we thought would be hardest to distinguish when developing our dataset. Further, the model vary rarely or never made mismatches between hazardously different question types such as MATH and LAW or CONV and CODE, where a misclassification would lead to a model which is very unlikely to respond effectively being delegated to respond on behalf of the ensemble. This lack of significant mismatches further shows to us that this model's accuracy is very satisfactory.

The confusion matrix for our final test performance can be found in Fig 8, the accuracy while training can be found in Fig 9, and label-specific accuracy can be found in Fig 10.

Table 4: Method 2 Model Hyperparameters

Hyperparameter	Value
Epochs	25
Learning Rate	0.01
Embedded Dimension	100
Hidden Dimension	50
Optimizer	SGD
Loss	CrossEntropy
Weight Decay	0.001
Momentum	0.9
LR Decay	x0.75 / epoch

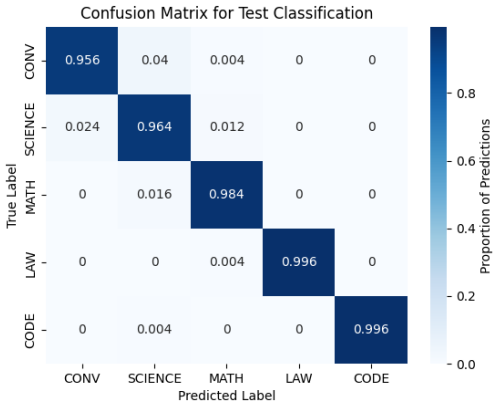

Figure 8: Method 1: Confusion Matrix

4 Discussion

Picking a model fine-tuned for specific types of questions is crucial to producing accurate outputs. In our ensemble, proper model selection enhances the overall performance by leveraging the specialized expertise within each model. Since these individual models excel in different question categories, the ensemble outperforms any single model on its own. This approach should not only lead to improved accuracy in our five designated domains but also increase the adaptability of our ensemble method to handle many types of user queries.

4.1 Key Findings

From the results of the models above, we can see that a larger model is required to properly capture all of the features of the data, reaching much stronger long-term performance with the 100 embedding dimension model than the 24 embedding dimension model. However, this result came at a drawback of training time and computational resources, making it difficult to test many different

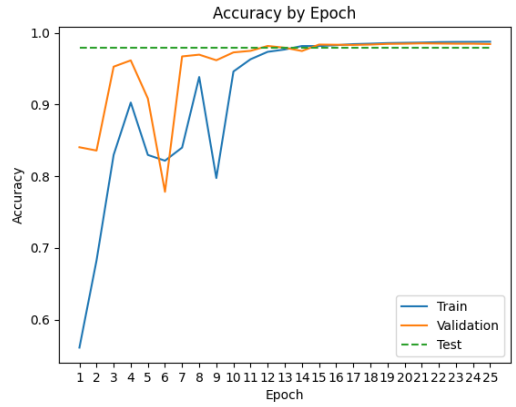

Figure 9: Method 1: Training Accuracy

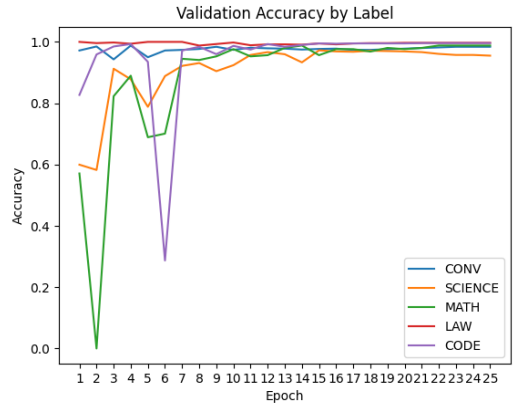

Figure 10: Method 1: Accuracy by Question Type

combinations of hyperparameters.

The importance of higher-dimensional word embeddings in the context of our results is threefold. First, more embedding dimensions allow the model to better learn complex patterns and associations in the data, which is especially beneficial when dealing with diverse domain-specific language (which is something we see both models learn quickly from the LAW data). Second, a larger embedding space allows the model to better capture contextual information to distinguish between domains based on context alone. For example, scientific texts typically include technical terms and references to scientific principles, while conversational texts usually involve informal language and varied sentence structures, which the models are able to capture more directly with larger embedding dimensions. Lastly, higher dimensions of word embeddings allows the model to simply retain more information and reduce information loss, creating a more detailed understanding of the different domains. However, this comes at the risk of overfitting, which is where the previously mentioned techniques of regularization come in to play, preventing the model from memorizing results of the training data.

4.2 Impacts of Misclassification

A natural byproduct of a classification based ensemble method like what we've developed is the risk of misclassification. Since expecting a machine learning model to achieve perfect test accuracy is unrealistic, the hazards and risks of misclassifications must be noted. "Minor" misclassifications such as those between largely similar question style or domains such as MATH and SCIENCE may not be very dramatic - since the fine tuning of those models to perform well in those domains may lend them to be competent (albeit less capable then the best model) at responding to prompts from adjacent categories, we anticipate that our ensemble would still outperform a single general model even in the case of minor misclassification. Major misclassifications such as between MATH and LAW offer a much larger risk however. A model fine tuned in law questions would be generally useless at responding to a math question and would likely perform equivalently to random guessing or an untrained model. In the event of major misclassifications, we expect our model to perform terribly - and much worse than a generalized model. Fortunately, major misclassifications like this are exceptionally rare with how high the performance of our model is which allows for this risk of terrible performance in the case of misclassifications to not be a significant concern with our model.

5 Division of Work

5.1 Joshua Symonds

- Pair Programming with Clark to:
  - Gather Training Data
  - Identify Best Models for Ensemble
  - Preprocess Data
- Develop Machine Learning Model Fundamental Architecture
- Trained Method 2
- Prepared slides for final Presentation
- Helped write the final report

5.2 Clark Kaminsky

- Pair Programming with Joshua to:
  - Gather Training Data
  - Identify Best Models for Ensemble
  - Preprocess Data
- Debug and refine Machine Learning Model Architecture
- Trained Method 1
- Prepared slides for final Presentation
- Helped write the final report

5.3 Ojaswi Bhimeni

- Loaded a pre-trained PyTorch model for text classification.
- Created a Flask web application to handle user interactions.
- Defined routes for rendering the main page and processing predictions.
- Implemented logic to preprocess input data and obtain predictions from the model.
- Mapped model predictions to human-readable class labels.
- Redirected users to specific URLs based on the predicted class.
- Set up debug statements to monitor the application's behavior.
- Ensured the proper initialization and running of the Flask app in debug mode.

6 Web Wrapper

The final wrapper developed for our ensemble method automatically routes the prompt to the best model, as shown in Fig 3 and Fig 4. As part of this effort, our UI was designed to directly navigate users to the relevant website based on the model's prediction. This approach aims to provide users with a quick and simple experience and with access to the information or services related to their input. Given more time with the project we would have liked first, to respond seamlessly within the application, and second, to develop an intelligent way to handle misclassifications, and third, develop a more aesthetically pleasing and interactive user interface for this model.

7 Conclusion

In the rapidly evolving landscape of Large Language Models (LLMs), users often interact with these models as all-knowing black boxes, expecting accurate responses to a wide array of questions without fully understanding the models' limitations. Specialized LLMs excel in specific domains and an ensemble of several offers better performance than a single general model, but their potential remains underutilized due to user unawareness or a lack of guidance on selecting the most suitable model for their questions.

To address this challenge, we have developed a delegation-based LLM ensemble approach in this paper. This method empowers users to engage with a unified interface for their conversational queries while harnessing the domain specialization of an ensemble of large language models. Our approach utilizes a machine learning delegator to identify the most appropriate LLM from the ensemble based on the prompt's question style and informational domain. Then, the most capable model responds to the prompt on behalf of the ensemble, enabling it to generate the most relevant response. By doing so, this ensemble approach enables users to access the collective expertise of a wide range of specialized LLMs without requiring them to manually identify and employ the best model for their specific task.

Our methodology aims to bridge the gap between user expectations and LLM capabilities, enhancing the overall effectiveness and user experience when interacting with large language models. Through our final product, we offer users a seamless and efficient means of accessing the capabilities of domain-specific LLMs without the burden of identifying or interfacing directly with a large variety of LLMs. This ensemble method should allow for user prompts to be responded to appropriately and correctly more frequently than if the user simply interacted with a single general LLM.

In conclusion, as LLMs continue to advance, ensuring that users can easily harness their capabilities becomes increasingly important. The delegation-based ensemble approach we developed serves as a step towards making LLMs more user-friendly and effective, facilitating access to their domain-specific prowess, and ultimately enriching the user experience.

Our final product correctly identifies the best large language model in its ensemble with 98% accuracy which dramatically outperformed out 70% accuracy benchmark for a successful model. Further, our model rarely makes catastrophic misclassifications which leads to a potentially more robust final product which avoids dramatically erroneous responses.

8 Future Work

The product of this work can be extended in several ways. Most obviously, future work can involve more styles and domains of questions which are tied to a new model in the ensemble. This allows for more granularity and specialization of models in the ensemble which will likely lead to further improving the performance of our design over a single general model. Specific examples of future prompt categories might include language translation, biology/medical, cooking, chemistry, history, and many more.

Another potential expansion of the work outlined in this paper would to further explore result mixing like what was done in our related works [4]. Instead of trivially selecting the best model - which risks entirely classifying a prompt - a priority can be assigned to each of the models in the ensemble based on their capability to effectively respond to the prompt. Then, a result can be generated for the ensemble by fusing the outputs of each model as well as its priority ranking; this can output a result representative of the entire ensemble instead of only the single model we expect to perform best.

Another anticipated improvement upon our model that could be explored in future work is improving its behavior under uncertainty. When classification between which model is ambiguous encoding some set of hierarchical default models may improve performance. For example, if a prompt is ambiguous between math and science the ensemble may perform better if it always chooses science if it is ambiguous. Further, if confidence does not surpass a certain threshold a catchall model such as GPT-4 may benefit the ensemble so it does not need to decisively act under uncertainty.

9 Github Link

Our Github Repository can be found here: `https://github.com/kaminsca/Ensemble-LLM`

References

[1] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv* abs/1803.05457. https://api.semanticscholar.org/CorpusID:3922816.

[2] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases.

[3] OpenAI Inc Github Inc. 2019. Github copilot - you ai pair programmer. https://github.com/features/copilot.

[4] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion.

[5] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems.

[6] Chenxiao Liu and Xiaojun Wan. 2021. Codeqa: A question answering dataset for source code comprehension.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

[8] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models.

[9] OpenAI. 2023. Gpt-4 technical report.

[10] Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge.

[11] Jaromir Savelka, Arav Agarwal, Christopher Bogart, and Majd Sakr. 2023. Large language models (gpt) struggle to answer multiple-choice questions about code.

[12] Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. Gpt can solve mathematical problems without a calculator.

[13] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset.