Hivemind: An Architecture to Amalgamate Fine-Tuned LLMs

Stanford CS224N Custom Project

Matthew Mattei
Department of Computer Science
Stanford University
mdmattei@stanford.edu

Matt Hsu
Department of Computer Science
Stanford University
matthsu@stanford.edu

Ramya Iyer
Department of Computer Science
Stanford University
ramya1@stanford.edu

Abstract

Large language models (LLMs) exhibit a wide variety of strengths, raising the question of whether a system can be constructed to harness the abilities of multiple LLMs in tandem. To address this, we developed Hivemind, an architecture comprising a classifier model and a network of other, specialized models. The classifier is trained to determine which model in the network is best equipped to process a given user input, subsequently passing the input to the selected model and relaying the output back to the user. We evaluated Hivemind's efficacy by finetuning pretrained models to specialize in answering questions from various subject categories (STEM, humanities, social sciences, "other") in our network. We additionally trained a classifier to pass multiple-choice question inputs to the appropriate specialized model. Our experimentation and results found that Hivemind performs significantly better than a non-finetuned vanilla model in terms of answering questions correctly. However, further alternations to the dataset and the fine-tuning process of the models in the network are necessary to fully demonstrate its potential.

1 Key Information to include

- Mentor: Ryan Li
- Team Contributions: All three team members contributed equally and to all aspects of the project, including data acquisition and formatting, finetuning, and the final project writeup. In terms of specific contributions, Matthew finetuned the classifier and network models and wrote scripts to calculate the scores of the models, Matt gathered and formatted training and evaluation data and wrote scripts for the random architectures, and Ramya ran early experiments, gathered and formatted training and evaluation data, and created the tables and graphs of results.

2 Introduction

A major strength of pretrained large language models is that they can be finetuned for various tasks. From the broad function of "question answering" to the niche specialization of "obtaining textual descriptions of the lyrics, melody, and musical components of various songs" Doh et al. (2024), there are endless possibilities of what models and frameworks can be created. One drawback of finetuning, however, is those models finetuned for very specific tasks may not perform strongly on tasks that do

not fall within that niche, while models finetuned for more general tasks require immense amounts of data to perform most effectively.

Multi-agented systems are a relatively new phenomenon in the field of machine learning. Broadly speaking, multi-agent systems enable a group of agents, or programs capable of perceiving the environment around them and operating accordingly, to collaborate and collectively perform better than they would have individually. One of the first and most well-known multi-agented systems is called Autogen, an architecture that enables agential customization and conversation Wu et al. (2023). Autogen's framework is unique in that it allows agents to provide feedback for one another and cooperate to improve. Similar research has further explored the capabilities of multi-agented systems.

On a finer level, multi-agented systems necessitate the implementation of *task classification*. They must be able to accurately determine what tasks should be assigned to what agent(s). In this study, we examine the effectiveness of this task classification by constructing a customizable multi-model architecture called *Hivemind* that, given a list of models, is able to determine which model an input should be passed to for optimal results. To test Hivemind's efficacy and determine what design implications should be kept in mind when plugging models into the architecture, we finetune several Llama3-8b models to be specialized in solving problems belonging to certain subject groupings: one for STEM, one for humanities, one for social sciences, and one for "other" (accounting, business, clinical studies, etc) Meta AI (2024b). To accomplish this, we utilize the Massive Multitask Language Understanding (MMLU) dataset, which consists of multiple-choice questions and answers in a wide variety of subjects Liang (2024). These finetuned models form our network. We then train a central Llama3-8b model to be our "classifier" model, finetuning it on the MMLU dataset to be able to classify questions as one of the above four categories Meta AI (2024b). Our results indicate that Hivemind performs significantly better than a non-finetuned baseline model but on par with the individual models in the network, suggesting that, while the finetuned classifier architecture has potential, certain guidelines must be followed to optimize potential.

3 Related Work

Llama 3 Due to the recent and rapid advancements in large language models (LLMs) such as ChatGPT and GPT-4, many major companies have begun exploring the potential of developing their own LLMs Achiam et al. (2023). Recently, Meta AI released Llama 3, which improved on reasoning from the previous versions of Llama Meta AI (2024b). As compared to the previous Meta AI Llama 2 model, the architecture of Llama 3 allows for increased efficiency in encoding language, greatly increasing the performance of the model Meta AI (2023). Through this model, we are able to fine-tune multiple versions for specific tasks, ideally enabling the model to be superior in its category.

Multi-Agent Systems As mentioned earlier, multi-agent systems involve a network of agents working together and communicating with one another to optimize task performance. Autogen distinguishes itself from other similar architectures due to its customizable and conversible agents, its conversation programming between agents, and its unique prompting techniques.Wu et al. (2023) It has shown remarkable accomplishments in the areas of math problem-solving, retrieval-augmented code generation, decision-making, and even conversational chess. Autogen's achievements are impressive, and their success in the field of multi-model systems is partially what's motivated us to explore that realm ourselves. Further papers have also explored the ideas of multi-agent systems in coordination games, enhancing text quality similar to human preferences, and problem solving Agashe et al. (2023); Chan et al. (2023); Rasal (2024).

4 Approach

We experimented with multiple approaches to select the models in our Hivemind architecture and determine appropriate datasets to train them on. Initially, we aimed to analyze Hivemind's propensity for creating a network of models designed to complete coding tasks. We selected several models finetuned for code generation and discussion, each with certain specialties such as code completion or Python programming. We aimed to finetune our classifier model route each input to the appropriate model. However, we encountered difficulties in locating data to finetune the classifier on as each model would often perform similarly, if not identically, on the same input. In our initial approach, we

ran inputs through all models in the network and manually labeled which outputs were strongest. This method proved to be slow and subjective. We also considered using an LLM-as judge but concluded that this method would also be susceptible to bias.

To solve this issue, we elected to create a network of models, each finetuned on a specific subject domain: STEM, humanities, social sciences, or "other". We trained our classifier model to determine which of these domain-specific models each input should be passed to. We utilized a dataset of multiple-choice questions labeled with both answers and subjects. This process, as detailed in Figure 1, enabled us to effectively train our classifier model.

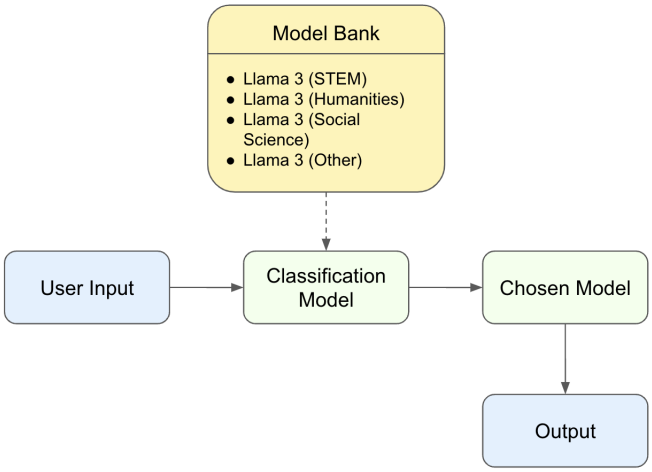

Figure 1: Overview of the Hivemind Architecture, detailing the flow from the User Input to the Classification Model which selects the model from the Model Bank to the Output.

4.1 Division of Subjects

The MMLU dataset contained questions from 57 different subjects. We organized them into four categories to distinguish the data each finetuned model would be trained on and denote the correct assignments for the classifier model. The four categories were STEM, humanities, social sciences, and "other". A comprehensive list of these divisions can be found in the Appendix (Figure 7).

4.2 Network Models

We finetuned four Meta-Llama-3-8b models on different portions of the MMLU dataset, one on each of the four categories marked above. Datasets were prepared by solely including each multiple-choice question and the corresponding answer, excluding any additional information. Due to resource and time constraints, each model was finetuned on roughly 3000 data points, except for the humanities model, which was finetuned on about 4,700. While this disparity was unintentional, subsequent results suggest that this alteration was insignificant. We utilized the following finetuning statistics for each of the individual models.

```
n_epochs = 4,
n_checkpoints = 1,
batch_size = 8,
learning_rate = 1e-5
```

We selected the default number of epochs when finetuning, as when analyzing checkpoints we deduced this was the best way to maximize performance without expending too much time and money. Due to the relatively small dataset size, we also wanted to prevent overfitting. For similar reasons, we also went with a relatively small batch size of 8 (the minimum for Llama-3 8b) and an

average learning rate of .00001. Selecting these more conservative parameters also helped us avoid memory overload.

4.3 Classifier Model

The classifier model was a Meta-Llama-3-8b model trained on the same questions as the network models. Rather than pairing each multiple-choice question with its respective answer, each multiple-choice question was paired with its respective category (STEM, humanities, social sciences, or other). This is because the classifier's job is to decide which model it should be passed to rather than respond to the inputs directly. It was trained on the same questions as above, meaning there were about 13,700 data points. Due to the dataset size not being larger by too much and the classifier model being the center of the architecture, we utilized the same finetuning statistics as listed above.

4.4 Pipeline

We finally constructed a pipeline that brings this architecture together. Users are prompted to provide an input, which is relayed to the classifier model. The classifier model then selects a model within the model bank to route the input, with the chosen model's output being relayed back to the user. This architecture is detailed in Figure 1. Future work on this project may involve further abstracting the Hivemind architecture to enable users to seamlessly add models to the network and finetune the classifier on a dataset themselves.

4.5 Baseline

For our baseline, we utilized a non-finetuned Llama-3-8b-hf model and evaluated its performance in correctly answering questions from the MMLU dataset. We wanted to see whether our architecture—passing to specific finetuned models—would be more effective than a general, pretrained but not finetuned model.

5 Experiments

5.1 Data

We used the Massive Multitask Language Understanding (MMLU) Hendrycks et al. (2020) dataset, splitting it into training and evaluation data so results were not tainted. This dataset contains 57 tasks on various subjects, which we separate into 4 main categories based on the MMLU paper. These 4 categories are STEM, humanities, social science, and other. Each dataset entry contains a question, a subject, a list of four answer choices, and the correct answer. The classifier maps question and answer choices to the corresponding subject, while the individually finetuned models map question and answer choices to the corresponding correct answer.

5.2 Evaluation method

We had two primary methods of evaluation. First, to evaluate the efficacy of the classifier, we passed 1000 questions from our evaluation dataset to the classifier, removing the subject labels from them and storing them elsewhere. We examined how often the classifier passed the question to the correct model (STEM, humanities, social sciences, or other), tracking additional statistics such as which models questions were passed to most or least often and which models received the highest proportion of questions that weren't meant for them.

Second, to evaluate the efficacy of the Hivemind architecture as a whole, we passed each of the same 1000 questions from our evaluation to the corresponding finetuned model that the classifier chose. We calculated how many of these questions the architecture answered correctly, comparing its performance in this metric to a number of other models or designs: passing the 1000 questions just to the baseline model described above, passing the 1000 questions to each the four models each individually finetuned on specific categories (STEM, humanities, social sciences, other), passing the 1000 questions to just our "mega-model" (a Meta-Llama-3-8b model finetuned on every question individually passed to the four other finetuned models), passing the 1000 questions to a random classifier architecture where the choice of which model to pass to is determined by random choice,

passing the 1000 questions to a random majority classifier architecture where three finetuned models are chosen at random to select an answer and the answer with the majority of votes is chosen (ties broken randomly), passing the 1000 questions to a majority classifier architecture where the answer with the majority of votes between all four finetuned models is selected (ties broken randomly), and passing the 1000 questions to a "perfect" classifier architecture, where the question is passed to the correct model every time (simulated by passing the question to the topic expert for the topic category the question fell under).

5.3 Experimental details

We wrote a script for both evaluation methods. We used the Together AI API to feed inputs to our classifier model, as well as our other models, and collected the outputs in a text file. These scripts additionally summarized the output results at the end and computed relevant numbers, listing how often the input was passed to the correct model (for the first evaluation method) and how often the correct answer was selected (for the second evaluation method). Each evaluation took roughly 40 minutes due to the model runtime and the length of API calls, which did somewhat limit our ability to run multiple evaluations.

5.4 Results

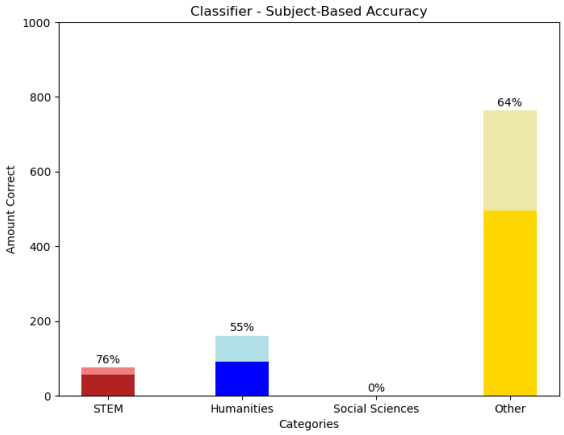

Figure 2: Classifier: Assignment Accuracy

For our first evaluation metric, we analyzed how often the classifier model passed a question to the correct individually-finetuned model (Figure 2). The classifier model successfully classified **384 out of 1000** questions, meaning it had a 38.4% success rate. While this is clearly better than random choice, which would have had a 25% success rate, this isn't quite ideal, as it means the classifier is passing to the incorrect model more often than not.

Clearly the classifier model passed to "other" far too often and the others not nearly enough, particularly social sciences. We suspected that the classifier was defaulting to choosing "other" whenever it wasn't immediately sure. To test this hypothesis, we ran a second experiment, renaming "other" as "niche" and utilizing some additional prompt engineering tools to attempt to get it to choose "other" less often, but the results were the same. We then ran a third experiment, removing the "other" category entirely and consolidating the subjects that fell into "other" into either STEM, humanities, or social sciences. Most notably, a category in "other" called "miscallaneous" occupied a large part of our evaluation dataset. Since these questions mostly consisted of arbitrary fun facts, we removed them entirely. After re-finetuning the classifier model on three categories instead of four and creating a new evaluation dataset (still with no overlap with the training dataset) with 334 STEM questions, 333 humanities questions, and 333 social sciences questions, we obtained the following results (Figure 3).

The model overall classified **572 out of 1000** questions correctly for a 57.2% success rate, which is a substantial improvement from before. We do have to account for there being fewer possible choices

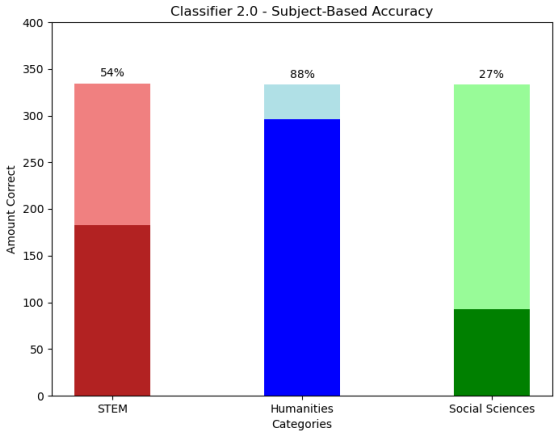

Figure 3: Classifier: Assignment Accuracy After Removing "Other" Category

this time, so we compare to random guess. The original classifier did $\frac{.384}{.25} = 1.546$ times better than random chance, while this classifier did $\frac{.572}{.333} = 1.718$ times better than random chance, which is still a noticeable improvement. Were it not for resource and time limitations, we would have re-finetuned and evaluated with these new categories and examined if other results changed as well.

For the second evaluation metric, we analyze how many answers various models and systems (including our Hivemind classifier system) got correct (Figures 4, 5, 6). Charts with more information, including breakdowns by subject, can be found in the appendix.

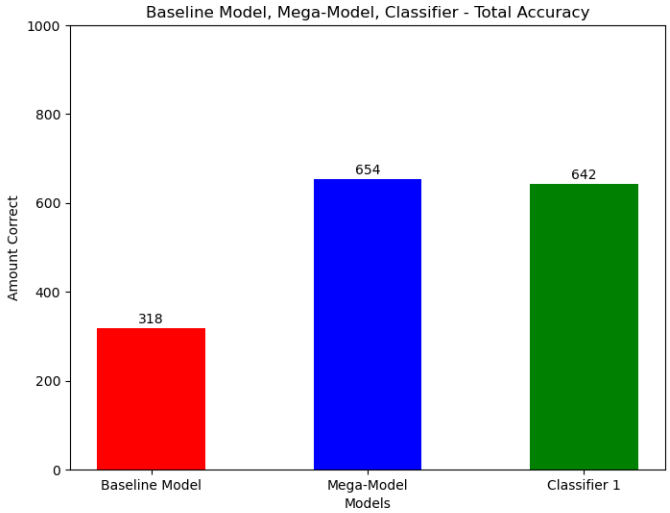

Figure 4: Baseline Model, Mega-Model, and Classifier System: Number of Questions Answered Correctly

We note that, aside from the non-finetuned baseline model, performances are roughly the same. All models and architectures get **between 640 and 660 out of 1000** answers correct except the finetuned STEM model, which only got 627 correct. Meanwhile, the baseline model was about half as good, getting only 318 answers correct. This suggests that finetuning is effective, but finetuning on individual subjects doesn't make a meaningful difference in results on other subjects. For example, the finetuned STEM model answered less STEM questions correctly than the humanities, social

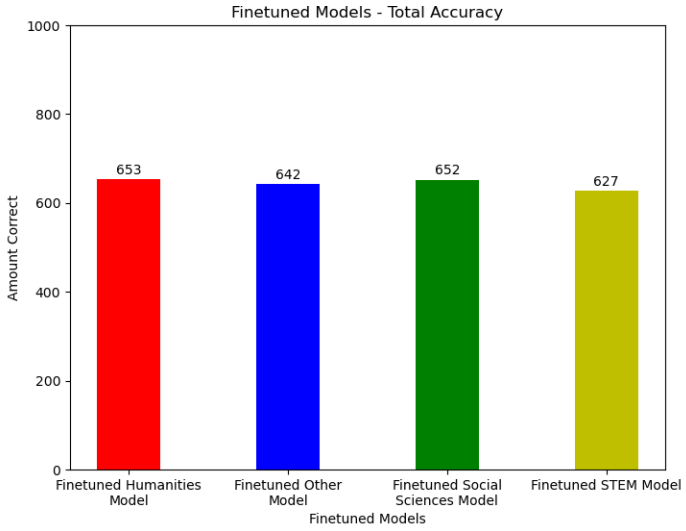

Figure 5: Individually Finetuned Models: Number of Questions Answered Correctly

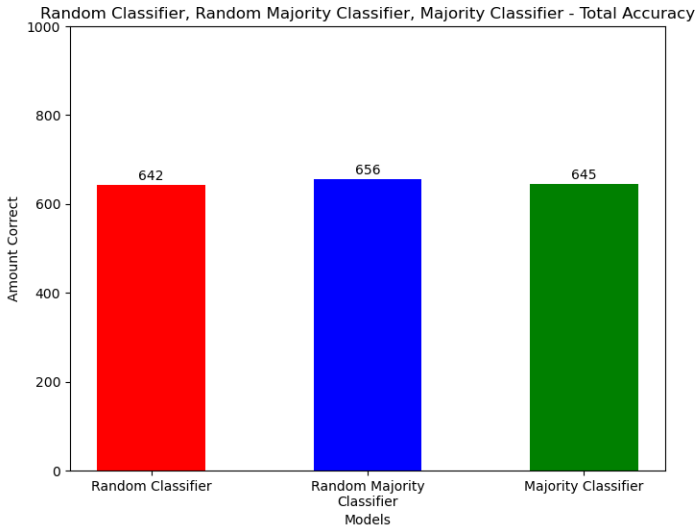

Figure 6: Random Classifiers: Number of Questions Answered Correctly

sciences, and "other" models (although it's worth nothing it just answered less questions correctly in general). It's probable that in the finetuning process, the individual subject models learned more about question answering in general rather than question answering for the particular subject.

6 Analysis

Given our results, we note that while classifiers can be effective, their efficacy is dependent on the distinction and the well-definedness of both the datasets and the models selected for the architecture. The vagueness of the MMLU categories is likely contributing to the classifier's errors. For example, an example MMLU question is:

Which of the following statements is TRUE concerning the standard regression model? A: 0: y has a probability distribution. B: x has a probability distribution. C: The disturbance term is assumed to be correlated with x. D: For an adequate model, the residual (u-hat) will be zero for all sample data points

The subject of this is "econometrics", which we count as a social science. However, given that this question is about mathematical modeling, one could reasonably assign it to the STEM category as well. Furthermore, there is some ambiguity with the categories themselves, particularly social sciences and humanities. Topics like sociology and religion fall somewhere in the middle, with the distinguishing factor being more *how* the subjects are studied rather than the material itself, something an LLM might have trouble picking up on.

That the mega-model and perfect pipeline models performed similarly to rest of the models (654/1000 and 653/1000 respectively) backs our proposition that the individually finetuned models are fairly similar. These two models should be strongest, as the former is finetuned on the entire training dataset spanning all four categories, while the latter passes each question to the model individually finetuned for questions of the same subject. However, both performed only marginally better than the rest. The Hivemind architecture thus seems to be most effective when the models in the network are fairly distinct—otherwise all inputs could just be passed to the same model. This raises an interesting balancing act, as if the models are all completely different (i.e. one for code generation, one for Chinese language conversation), then inputs can likely be filtered deterministically, with no need for machine learning. Conversely, if the models are all very similar, then we get the problem described above. A potential solution here is to finetune the models in our network on more data to make them more specialized, which is something we hope to do in the future.

7 Conclusion

Overall, we succeeded in our mission of constructing a multi-model architecture that utilizes a classifier model to feed inputs to different specialized models. While the architecture's performance as demonstrated above wasn't as successful as we had hypothesized, it was certainly not bad and indeed outperformed our baseline. Perhaps most importantly, we made several discoveries pertaining to our Hivemind architecture and how to best optimize its performance, namely

1. It is important for the models in the network to be sufficiently distinct; otherwise, the classifier's importance is lessened

2. When including an "other" or "default" model in the network, it's important to consider that the classifier will likely pass to it very frequently

3. When examining a training dataset for a classifier, deciding which model each input "should" pass to, i.e. the ground truths, may be challenging. If the dataset is not pre-labeled in that regard (or if the labels seem unreliable), it may be helpful to try techniques like crowdsourcing opinions or using LLMs as judges, though both these strategies have their drawbacks.

Our work suffered from certain limitations, the most prominent being resource constraints. By the time we completed all training and evaluation, we were down to $5 in Together AI credits, having spent nearly all of what we were granted. Similarly, we were constrained by time, as we had to finetune and evaluate multiple models, which we did several times. Because of this, both our classifier model and our individually finetuned models are not as potent as we would have liked. Another limitation is that sorting questions into STEM, humanities, and social sciences can be subjective, as discussed above, and our results would likely be different had we categorized differently. For future work on this project, we would like to help rectify these issues by finetuning each of our models for much longer periods of time on much larger datasets. This would hopefully result in not just a stronger classifier, but more specialized individual models in the network. Furthermore, we'd like to go through the training and evaluation dataset more carefully and divide the subjects into more granular categories. This would help remove some of the ambiguity from our category definitions and make our results more robust.

8 Ethics Statement

We utilize the pretrained Llama3 model in our architecture, developed by Meta AI. Meta AI did put effort and thought into developing responsible models to promote safety in deployment; through their Llama Guard editions and Code Shield, Meta bolstered safety and security for code suggestions, a common apprehension when it comes to LLMs Meta AI (2024a). However, there are still concerns. Llama3 was pretrained on "15 trillion tokens of data from publicly available sources", including "publicly available instruction datasets, as well as over 10M human-annotated examples" AI@Meta (2024). While Meta used tools like NSFW filters to screen their data and protect users, Kili (2024), there may still be bias present in the dataset it was trained on, and Meta either did not take substantial steps to account for this bias or did not put this information on their model card. This is problematic because pretraining models on datasets that contain bias introduces bias into the models. For example, if an LLM is pretrained on datasets that only refer to doctors as "he", it may grow to assume women cannot be doctors as a result of gender bias. Because Hivemind utilizes numerous Llama3 models, any bias contained in the Llama3 models is carried over to our architecture as well. One mitigation strategy for this would be to replace our Llama3 models with a model that is perhaps not quite as powerful, but has taken explicit steps in its pretraining to reduce bias. We could alternatively keep the Llama3 model but introduce a finetuning cycle to reduce bias. Significant research has already been conducted on this topic, such as the 2023 paper "Disclosure and Mitigation of Gender Bias in LLMs", and we can utilize it to reduce bias in our own system Dong et al. (2024).

A second potential instance of bias is related to the first but has more to do with the abstraction of our model. In our architecture, the pipeline between the classifier model and the network models is abstracted away, meaning the end user of any system built using it would simply provide an input and receive an output. All information about the process under the hood would likely be hidden. This has the potential to be problematic because a developer could hide some LLMs within their Hivemind network that are inherently biased or dangerous and could target vulnerable populations. For example, a developer could insert a model into their Hivemind that targets elderly users and attempts to get them to input their social security numbers, using the classifier to detect when this is possible. Additionally, developers could reveal some of the models in their network to lull the user into a false sense of security while hiding the biased or dangerous ones. A mitigation strategy for this could be to create a reporting platform/system for users of any systems built on Hivemind to report dangerous or biased content they find, whose system it came from, and how they found it, enabling users to better hold developers accountable and to make investigations into Hivemind systems easier. This would ensure that people's use of the Hivemind architecture is monitored and anyone attempting to use it for nefarious activity more easily found and stopped.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Saaket Agashe, Yue Fan, and Xin Eric Wang. 2023. Evaluating multi-agent coordination abilities in large language models. *arXiv preprint arXiv:2310.03903*.

AI@Meta. 2024. Llama 3 model card.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

SeungHeon Doh, Minhee Lee, Dasaem Jeong, and Juhan Nam. 2024. Enriching music descriptions with a finetuned-llm and metadata for text-to-music retrieval.

Xiangjue Dong, Yibo Wang, Philip S. Yu, and James Caverlee. 2024. Disclosure and mitigation of gender bias in llms.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Kili. 2024. Everything you need to know about meta's new model and its data.

Kaizhao Liang. 2024. MMLU Auxiliary Trained Set Labelled by e5-mistral-7b-instruct. https://huggingface.co/datasets/kz919/mmlu-auxiliary-train-e5-mistral-7b-instruct. Accessed: June 4, 2024.

Meta AI. 2023. Llama 2. Accessed: 2024-06-06.

Meta AI. 2024a. CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models. Accessed: 2024-06-06.

Meta AI. 2024b. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI Blog*. Accessed: 2024-06-06.

Sumedh Rasal. 2024. Llm harmony: Multi-agent communication for problem solving. *arXiv preprint arXiv:2401.01312*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

A Appendix

This appendix provides further detailed information and results from our experiments. It includes more comprehensive divisions of the MMLU dataset as well as additional comparisons that we plot and found.

A.1 MMLU Datasets

Figure 7 and Figure 8 offer a more detailed breakdown of the subjects and categories found in MMLU research, as reference in Hendrycks et al. (2020). Figure 7 presents the subjects and categories exactly as provided in the original paper, Hendrycks et al. (2020). In contrast, Figure 8 reorganizes the "Other" category to organize within the three domain-specific categories: STEM, Humanities, and Social Sciences.

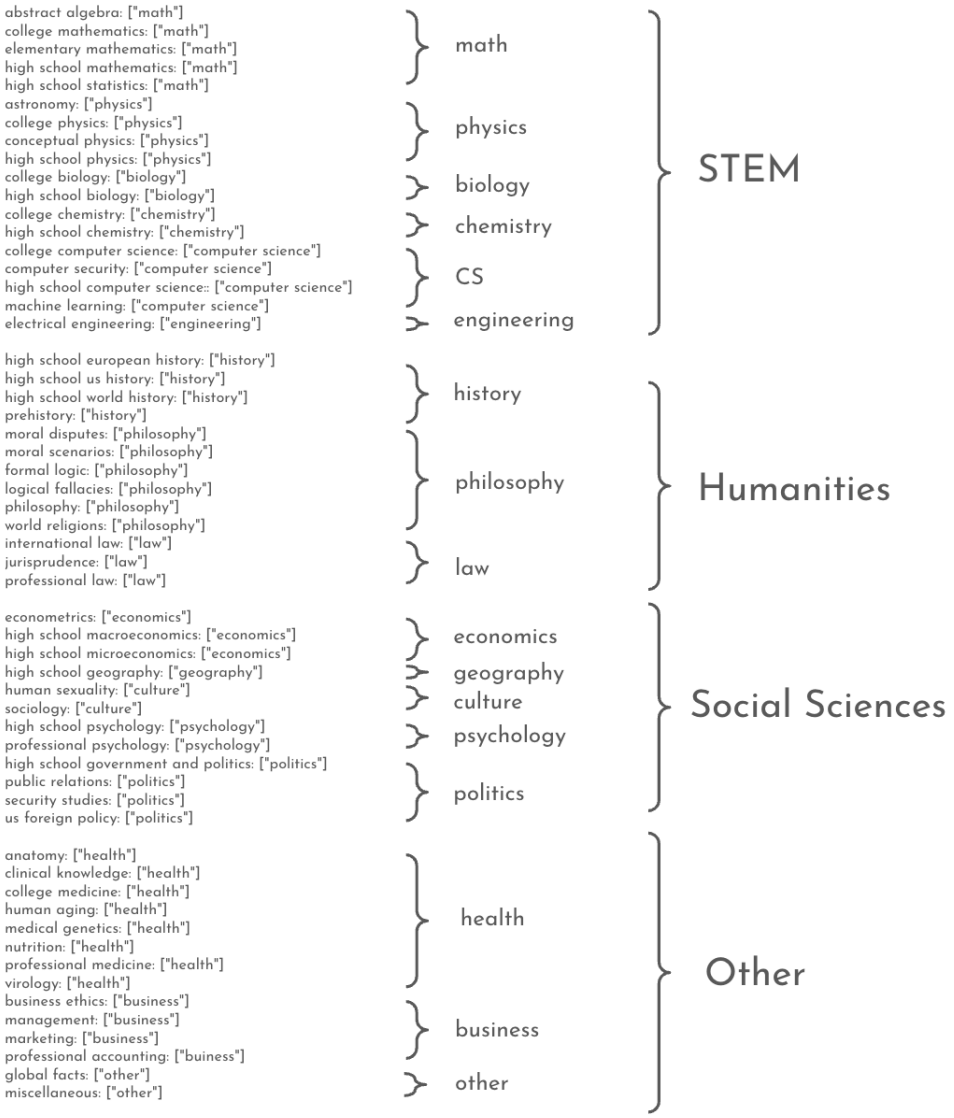

Figure 7: Categorization of MMLU Subjects into Subject Categories: This figure illustrates how we divided the MMLU subjects into our four categories.

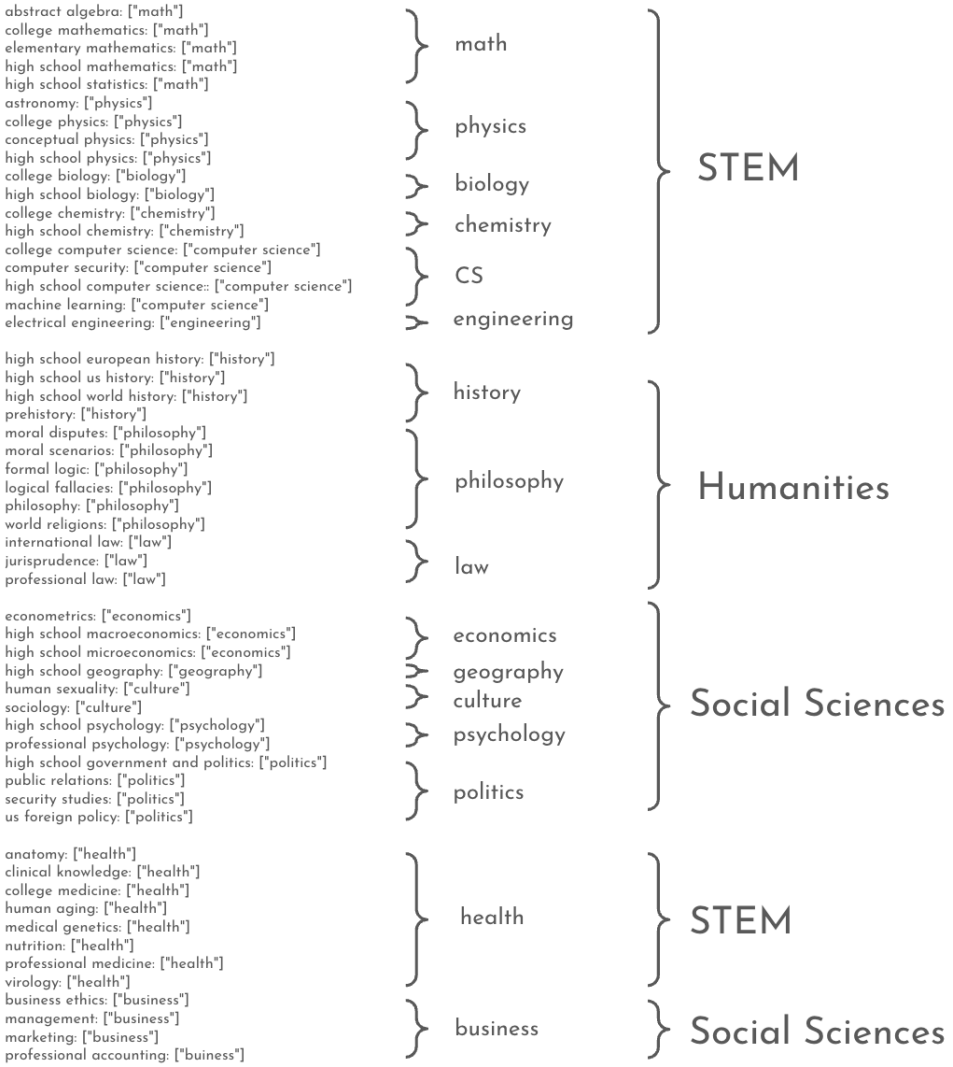

Figure 8: Categorization of MMLU Subjects into Subject Categories Without "Other": This figure illustrates how we merged the "other" category into the other three for the MMLU dataset.

A.2 Baseline Model and Mega-Model: Subject-Based Accuracy

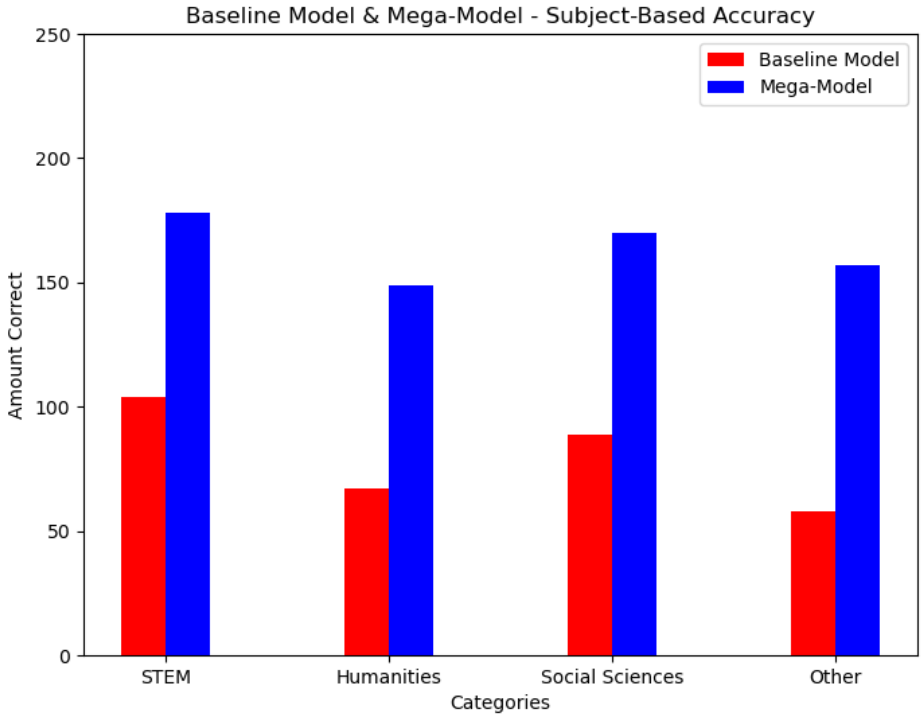

Figure 9: Baseline and Mega-Model Subject-Based Accuracy: This figure illustrates how many questions the baseline model and the mega-model got correct by subject. Recall the baseline model is a non-finetuned Llama3-8b model, while the mega-model is a Llama3-8b model finetuned on all four subject categories and not just one. Note that there were 250 total questions in each category.

A.3 Finetuned Models: Subject-Based Accuracy

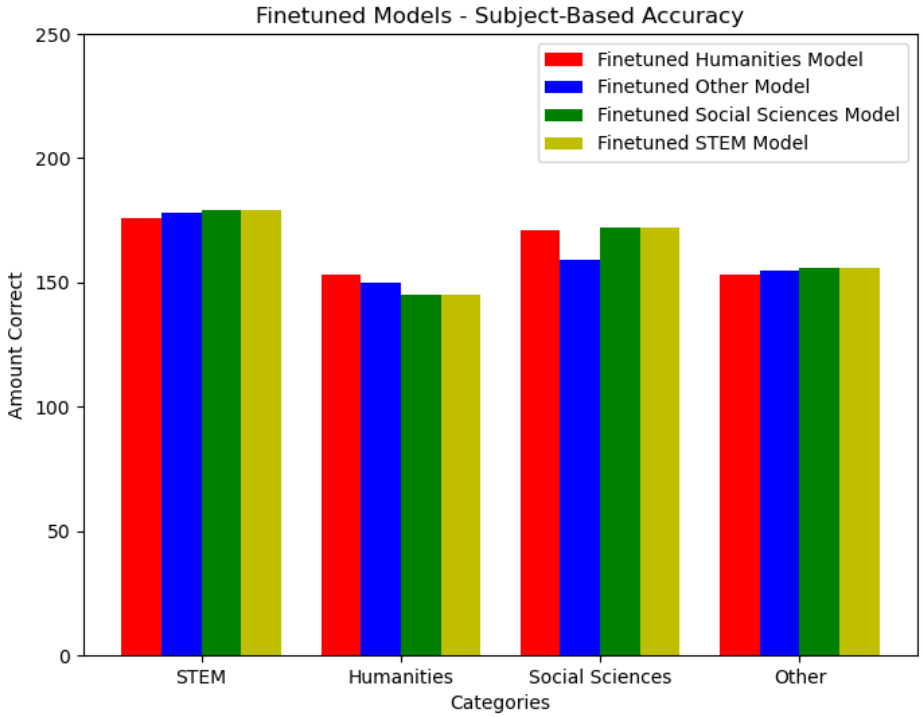

Figure 10: Finetuned Models Subject-Based Accuracy: This figure illustrates how many questions each of the individually finetuned models in our network (one for STEM, one for humanities, one for social sciences, and one for "other") got correct by subject. Note that there were 250 total questions in each category.

A.4 Random Classifier, Random Majority Classifier, Majority Classifier: Subject-Based Accuracy

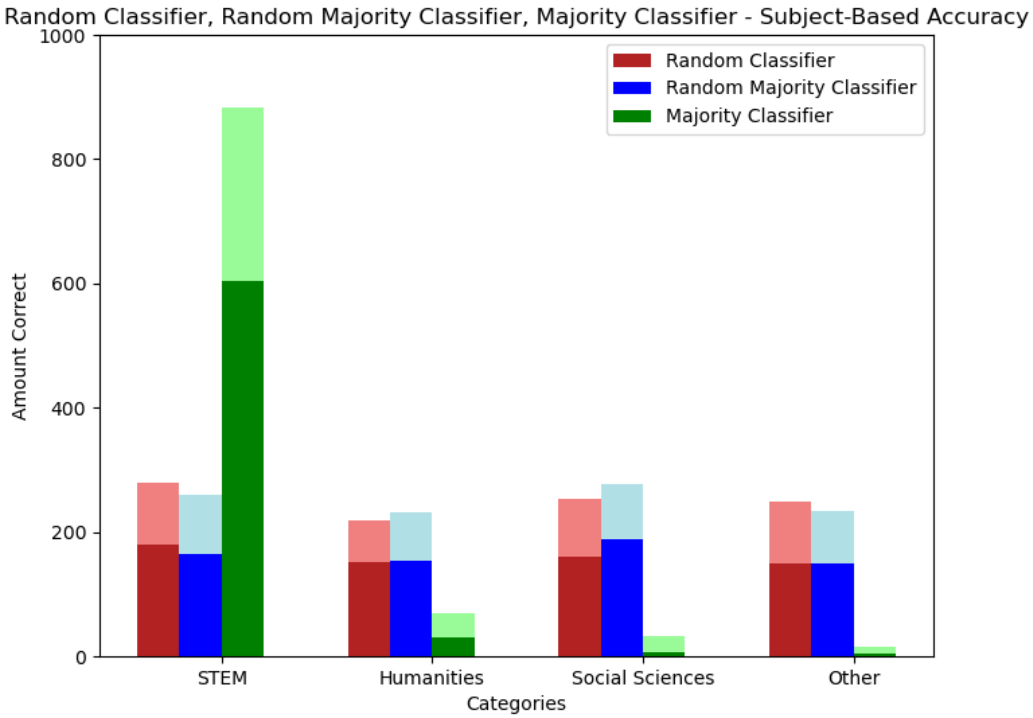

Figure 11: Random Architectures Subject-Based Accuracy: This figure illustrates how many questions the three random-based architectures got correct by subject. The random classifier replaces the classifier in our architecture with random choice. The random majority classifier replaces the classifier in our architecture with choosing three models at random and picking the majority answer among them, with ties broken randomly. The majority classifier is the same, except it picks the majority among all four models instead of just three.

A.5 Perfect Pipeline: Subject-Based Accuracy

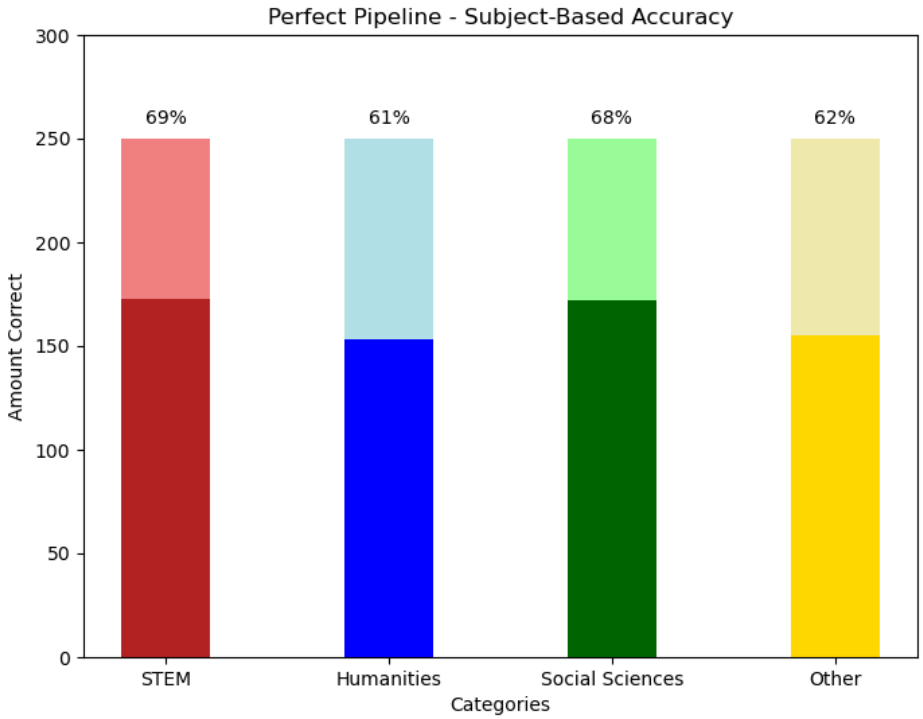

Figure 12: Perfect Pipeline Accuracy: This figure illustrates how many questions the our classifier would have gotten correct by subject if it passed each input to the correct model. Note that there were 250 total questions in each category. In total, the "perfect" classifier would have achieved a score of **653**, which is not far off the results of the actual classifier.