Adapters for hyperspecific tasks: Small-scale LLMs writing better API calls

Anonymous ACL submission

Abstract

MoE models and other complex routing methods have garnered a lot of attention. Procuring those models is a costly endeavour, which is complex to distribute. This paper shows that simple hard routing has a lot of potential in Tool-Use QA applications, allowing for more distributed training and decentralization, and has potential applications elsewhere. The task considered is based on a proposed with separated the ReACT stages for separately trained models: Planner, Caller and Summariser. Previous approaches have largely relied on zero-shot comprehension of the nature of specific APIs based on the provided documentation. Zero-shot abilities of small models are limited and studies show that most commonly failures occur at the Caller stage of the pipeline. Therefore this study shifts away from zero-shot assumptions by using a hard routing-based strategy utilizing expert adapters for each category of APIs. The experimentation has shown that this pipeline can allow the 7 Billion model, to beat much larger, modern and closed-source models used in a zero-shot scenario on this task.

1 Introduction

The growth in computational and architectural capabilities has enabled the size expansion of Large Language Models (LLMs), allowing new capabilities to emerge (Brown et al., 2020; Wei et al., 2022). Amongst those previously infeasible tasks is Tool-Use (Hsieh et al., 2023). In their review, Wang et al. (2024) describe it as providing the LLM with various utilities that are not part of the neural model, allowing the pipeline to perform previously impossible actions and improve the quality of the answer. Those actions can range from consulting a calculator to achieve better numerical answer precision (Schick et al., 2023), using a web browser to fetch current information that was not present during the training of the LLM (Zhuang et al., 2023), or even controlling an external robotic assistant through calls (Vemprala et al., 2023).

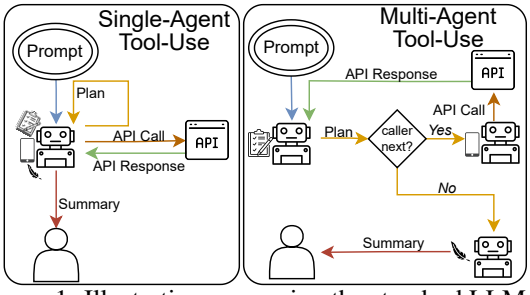

Figure 1: Illustration comparing the standard LLM assistant Question-Answering approach with the Tool-Use ReACT-based pipeline (Yao et al., 2023), in both Single-Agent (Qin et al., 2023) and Multi-Agent (Shen et al., 2024) configurations.

Current solutions based on small models, such as the 7B Llama model used in this study, often fail due to incorrectly generated API calls. Xu et al. (2023) highlight that API call malformation in the shape of incorrect Argument Population[1] is the most significant contributor to failures of the entire pipeline. Therefore improvements made at this stage of the pipeline are highlight likely to improve the pipelines overall. Furthermore, other studies show the poor zero-shot abilities of the small models used in this pipeline (Wei et al., 2022), hence pointing to a hypothesis where a shift away from the assumption that zero-shot API comprehension is feasible, would potentially have a large positive impact.

2 Related Work

Multi-Agent Tool-Use. Shen et al. (2024) have considered that performing **ReACT** reasoning might be too complex for a single small model. Therefore, to reduce the reliance on this weak assumption, they fine-tune separate models on three distinct tasks: **Re**asoning (also referred to as Plan-

[1] Argument Population is the task of selecting the parameters and values for those parameters in the API call.

ning), **Act**ing (Calling), and Summarizing. This scheme allows the individual models to focus more on their respective tasks, which is helpful given their constraints.

While routing during training is a trivial problem, as the correct agent alignment is known apriori, during inference routing is a significant problem, as it is not apparent which agent should respond at each step. Therefore, Shen et al. (2024) trained the Planner to conclude their responses with `Next: caller` or `Next: summariser`, which is used to route the pipeline at each turn to the correct expert. The remaining experts have a simpler routing scheme, the Planner follows every Caller action, while all responses from the Summarizer conclude the turn. Figure 1 includes a simplified illustration comparing the two approaches.

Progressive Fine-Tuning. Comprehension of the overall conversation and task is crucial to every step in the pipeline; therefore, the authors of the Multi-Agent scheme argue that simple fine-tuning right from the base models might create models that are overly specialised in performing their assigned task, therefore lacking comprehension of the overall task. This prompted the authors to propose Global-to-Local Progressive Fine-Tuning (GLPFT). This scheme approaches the issue mentioned earlier by first fine-tuning a 'backbone' model on the entire task, where the single agent has to perform all three roles in the reasoning, and then using this model to fine-tune three models to take on specific roles. This approach aligns with the conclusion presented by Gururangan et al. (2020), who show that a gradual multi-step alignment to the target task yields better performance than direct fine-tuning directed at the task. In this way, GLPFT indulges in applying the transfer learning principle twice through the aforementioned Global-to-Local Progressive Fine-Tuning scheme, as the first 'Global' stage is applied on a Llama model already pretrained on Language Modeling. The hierarchical structure of the problems considered allows for stacking and fine-tuning LoRA adapters that are increasingly more specialised, inspired by the stacked adapter approach used by Gema et al. (2024) in the clinical domain. The adapters trained in the study are applied to the Caller model in turn indulging in further Progressive Fine-Tuning.

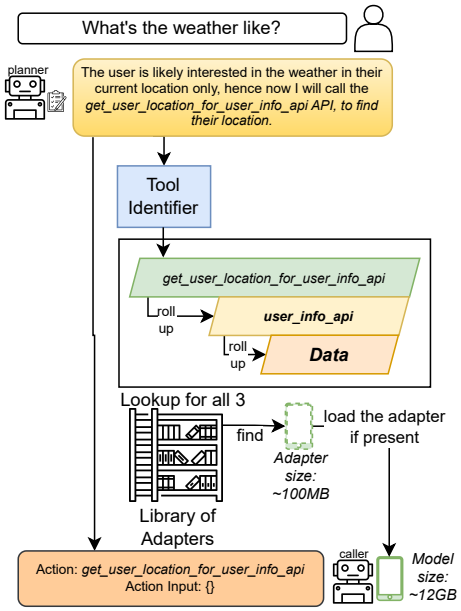

Figure 2: Illustration of how the plan determines the adapter choice in the pipeline.

3 Hard Routed Experts

The hypothesis investigated is that each endpoint could be regarded as a distinct task, warranting the use of a separate adapted set of parameters for each. Similarly, entire API families[2] can be treated as a single unit, and categories[3] comprising lists of API families can also be viewed as a unified task. This approach reduces the emphasis on the model's ability to learn API usage solely from the documentation in the prompt, as it allows the model to gain more insight by training on a dataset focused on a specific subset of the overall data. This hypothesis aligns with previous research suggesting that smaller models perform better when reliance on in-context learning is minimised through fine-tuning (Mosbach et al., 2023).

In a practical scenario, procuring even a relatively small subset of Llama-2 7B (Touvron et al., 2023) experts through fine-tuning becomes infeasible, as storing an entire model for each expert would be highly inefficient in terms of storage. To address this, adapters—specifically Low-Rank Adapters (LoRA; Hu et al., 2021)—were used due to their wide adoption in the NLP field and the fact that they allow for dynamic swapping of the adapters into the network. The latter consideration

[2]API families are sets of endpoints from the same route; for example, all endpoints accessible from `https://soundcloud4.p.rapidapi.com/` constitute the `soundcloud` API family.

[3]The creators of each API family annotate them by selecting one category from a list of 49 defined by RapidAPI.

is especially important for making this approach viable for practical scenarios.

Hard routing with a library of LoRA adapters is used, based on the description by Ostapenko et al. (2024), where the pipeline dynamically swaps one adapter onto the base model based on the endpoint names mentioned in the Planning step. In this scheme, backing off to the base version of the model is efficient, as it simply requires deactivating all of the adapters. This scheme has been shown in Figure 2. The parameter overhead of this solution is in the order of hundreds of megabytes per adapter, only one of which is used during inference at a time. This means that this solution does not significantly impact the inference requirements of the system, mostly requiring additional storage.

4 Dataset

Training data used in this study is the ToolBench dataset, which was a part of the Toollama study (Qin et al., 2023). Each expert is trained on a subset of tool calls that are relevant to the task split considered. This way, the agent can focus and derive expertise in their specific task. Therefore, the dataset was grouped by the tool used in each instance on an endpoint, API family, and category level.

This split points to the fact that, in the described pipeline, every classifiable plan can be answered by one of three models: the specific endpoint trained, the API family, or the category-wide expert. In cases where no expert is available (for instance, it was not trained yet), backing off to the base caller model is also possible.

5 Experiments

Metrics. The metrics considered are the exact match score for tool selection accuracy and the F1-score, which balances precision and recall, to evaluate the correctness of the arguments populated in the API call, including both keys and values. To emphasize the adapter portion of the pipeline, results in most experiments are grouped based on the output of the Tool Classification.

Heuristics. Initial experimentation focused on selecting the right level of abstraction and division between adapter tasks. It revealed that training API-wide experts can match the performance of procuring multiple endpoint-focused experts in most cases, with this broader and cheaper approach sometimes outperforming the latter approach. Training Category-wide adapters was not as successful as training API-wide adapters but still offered significant improvements. Predictably, larger categories (and often large APIs) proved to offer little to no benefit over the basic non-adapter approach, as can be seen in Figure 3. This shows that the shift in reliance on zero-shot adaptability has a large performance impact on the resulting pipeline, as the experts trained on larger task divisions have to be more flexible to zero-shot adaptation.

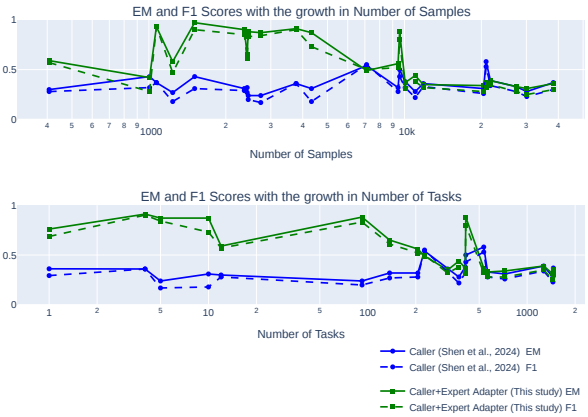

Figure 3: The effect of increasing the number of tasks (distinct endpoints) and samples on pipeline performance compared to the baseline.

Main experiments. To reduce potential biases and stochasticity the main comparison experiments were conducted on adapters trained based on the RapidAPI categorisation, despite the API-wide approach showing superior performance in the initial experiments. In addition to comparing to the non-boosted Caller model, the model was compared to two highly capable closed-source models attempting the task in a Zero-Shot and Few-Shot prompting scenario. The training required a single A100 GPU and potentially could be conducted on a less powerful GPU. The LoRA adapters were applied to the Q and V projections ($r = 16$).

In-Domain Tools. When considering tools from the ToolBench (Qin et al., 2023) TestSet, which covers the same tools as the training set used, the benefits are clear and very large for many categories. The results of this experimentation in terms of Exact Match are shown in Figure 4. More importantly, the Argument Population F1-score has also seen a great improvement over the baseline for the majority of the categories considered, as shown in Figure 5. For many of the categories, the proposed pipeline was able to outperform the closed-source models due to their expertise in that domain, which was previously infeasible due to

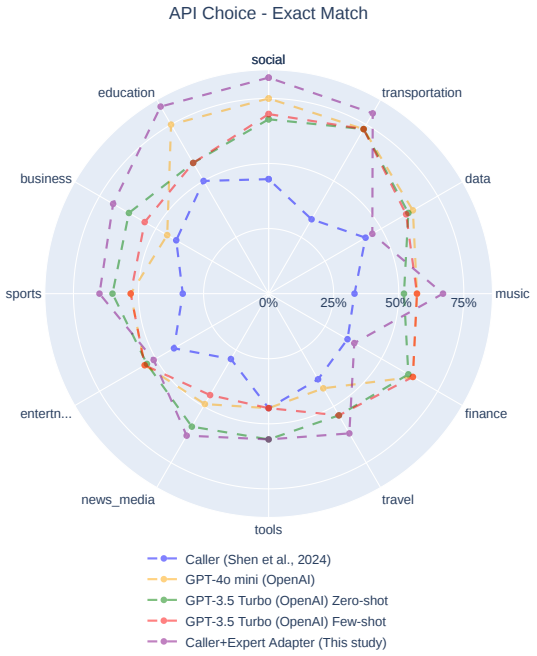

Figure 4: ToolBench test results for Callers, GPT models, and Experts. The metric considered measures how well the model was able to select the correct API.

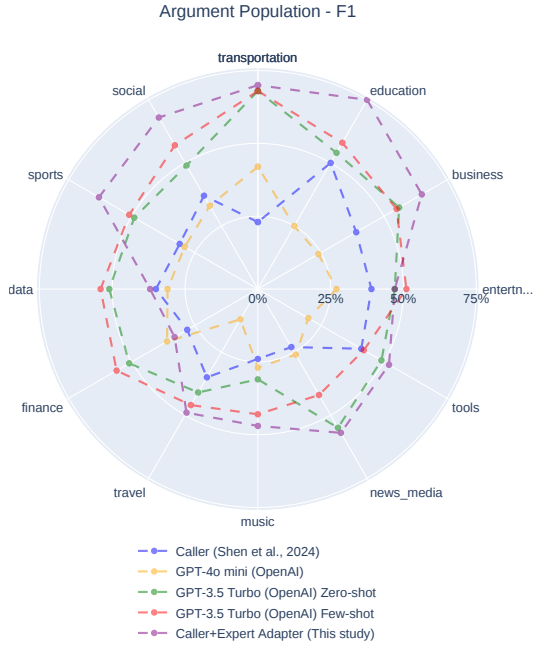

Figure 5: ToolBench test results for Callers, GPT models, and Experts. The metric considered measures how well the model was able to populate an object with relevant parameters to the tool.

their limited capabilities.

Out-of-domain Tools. The ToolAlpaca Test set (Tang et al., 2023) was used as a set of out-of-domain tools, not represented in the training set. The APIs were matched to the ToolBench API splits using zero-shot annotation by GPT-4. The expertise derived from only the training set and the reduction in zero-shot assumption could theoretically harm the performance of the pipeline in those scenarios. Table 1, goes contrary to this belief as the model performed well, sometimes even outperforming the baseline. This shows that the experts have acquired generalizable knowledge of those categories.

API Choice: Exact Match			
Category	Caller	Caller+Expert	Δ
Music	0.22	0.67	**+0.45**
News Media	0.46	0.46	
Business	0.40	0.60	**+0.20**
Finance	0.75	0.75	
Entertainment	0.15	0.15	
Education	0.38	0.44	**+0.06**
Tools	0.51	0.49	-0.02
Data	0.56	0.56	
Sports	0.00	0.00	
Social	0.50	0.50	
Argument Population: F1-Score			
Category	Caller	Caller+Expert	Δ
Music	0.11	0.39	**+0.28**
News Media	0.46	0.46	
Business	0.10	0.20	**+0.10**
Finance	0.38	0.56	**+0.18**
Entertainment	0.15	0.15	
Education	0.34	0.41	**+0.07**
Tools	0.47	0.47	
Data	0.44	0.39	-0.05
Sports	0.00	0.00	
Social	0.38	0.38	

Table 1: Performance Comparison Caller+Expert Adapter pipeline against the baseline (Shen et al., 2024) on an out-of-domain test set (Tang et al., 2023).

6 Conclusions

This study demonstrates the viability of an expert adapter pipeline for enhancing API call generation, building on the work of Shen et al. (2024). Reducing reliance on in-context learning significantly improves performance, allowing the small model pipeline to beat much larger closed-source models. Experts focusing on fewer than 200 endpoints consistently outperform the Caller model, while broader experts struggle to surpass the baseline. Aggregating endpoints into API family experts proves effective, reducing computational overhead for frequently modified APIs. Progressive fine-tuning also boosts overall performance, particularly when applied in multiple stages. The study highlights that partial implementations of Tool-Use, focused on challenging or popular APIs, can further improve results. These experts can be trained using lightweight, dynamically loaded adapters, which could even be distributed by API creators, amplifying the practical benefits of this approach.

7 Limitations

The study only considered the use of a specific moderately sized language model (namely Llama-2 in the 7B variant) in line with the research previously conducted on the same task. It is likely that using a different model for this task would make the entire pipeline perform better in practical applications.

Grouping tasks based on the categories taken from RapidAPI is another limitation of this study as the categories were assigned by the users of the service at the creation of the API. Some categories were overly broad (such as the Data category) leading to poor performance on the APIs from those sets. Tasks splits that consider the conclusions of this study would likely produce better expert than what is presented in this study.

Uncertainty in tool classification is a major limitation of this study. With only 64.3% accuracy in matching plans to the correct tool, and frequent mismatches between API calls and their classifications in the dataset. Enhancing the expert selection process or training the Planner to explicitly indicate the required tool in its output could help mitigate this problem. Additionally, GPT-4o mini's poor performance in the Argument Population, although not deeply explored in this study, raises concerns. While its lower cost makes it appealing for inference and data generation, its current results suggest limited viability unless these issues can be addressed.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Aryo Pradipta Gema, Pasquale Minervini, Luke Daines, Tom Hope, and Beatrice Alex. 2024. Parameter-efficient fine-tuning of llama for the clinical domain. *Preprint*, arXiv:2307.03042.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *Preprint*, arXiv:2004.10964.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *Preprint*, arXiv:2308.00675.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *Preprint*, arXiv:2305.16938.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. *Preprint*, arXiv:2405.11157.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small llms are weak tool learners: A multi-llm agent. *Preprint*, arXiv:2401.07324.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *Preprint*, arXiv:2306.05301.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan,

Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Preprint*, arXiv:2306.17582.

Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. 2024. What are tools anyway? a survey from the language model perspective. *Preprint*, arXiv:2403.15452.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *Preprint*, arXiv:2305.16504.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Preprint*, arXiv:2306.13304.

A In-domain tests results as a table.

Task	Caller (Shen et al., 2024)	GPT-4o mini (OpenAI)	GPT-3.5 Turbo (OpenAI) Zero-shot	GPT-3.5 Turbo (OpenAI) Few-shot	Caller+Expert Adapter (This study)
API Choice - Exact Match					
music	0.33	0.57	0.52	0.57	0.67
news_media	0.29	0.49	0.59	0.45	0.63
business	0.41	0.45	0.62	0.55	0.69
finance	0.35	0.64	0.62	0.64	0.38
entertainment	0.42	0.55	0.54	0.55	0.51
education	0.50	0.75	0.58	0.58	0.83
travel	0.38	0.42	0.54	0.54	0.62
tools	0.44	0.44	0.56	0.44	0.56
data	0.43	0.64	0.62	0.61	0.46
sports	0.33	0.53	0.60	0.53	0.65
transportation	0.33	0.73	0.73	0.73	0.80
social	0.44	0.75	0.67	0.69	0.83
Argument Population - F1					
music	0.24	0.27	0.31	0.43	0.47
news_media	0.23	0.26	0.55	0.42	0.57
business	0.39	0.24	0.56	0.55	0.65
finance	0.28	0.36	0.51	0.56	0.33
entertainment	0.39	0.27	0.47	0.51	0.47
education	0.50	0.25	0.54	0.58	0.75
travel	0.35	0.12	0.41	0.46	0.49
tools	0.41	0.20	0.49	0.42	0.52
data	0.35	0.31	0.51	0.54	0.37
sports	0.31	0.29	0.49	0.51	0.63
transportation	0.23	0.42	0.68	0.68	0.70
social	0.37	0.33	0.49	0.57	0.68

Table 2: Scores for various tasks based on different models: API Choice - Exact Match and Argument Population - F1. The models include Caller (Shen et al., 2024), GPT-4o mini (OpenAI), GPT-3.5 Turbo (OpenAI) Zero-shot, GPT-3.5 Turbo (OpenAI) Few-shot, and Caller+Expert Adapter (this study).