CSE291A - Large Model Reasoning Project Report

Mrinaal Dogra
Department of Computer Science
University of California San Diego
`mdogra@ucsd.edu`

Siddhant Lad
Department of Computer Science
University of California San Diego
`silad@ucsd.edu`

Krithika Iyer
Department of Computer Science
University of California San Diego
`k3iyer@ucsd.edu`

Harshit Timmanagoudar
Department of Computer Science
University of California San Diego
`htimmanagoudar@ucsd.edu`

1 Introduction

Large Language Models (LLMs) have emerged as a transformative technology across diverse industries, demonstrating remarkable capabilities in natural language understanding and generation. Despite their potential, significant challenges persist in adapting these models to domain-specific tasks, particularly those that require sophisticated interactions with external tools and data sources. Traditional fine-tuning approaches [1][2], while powerful, are often resource-intensive, computationally expensive, and challenging to scale across the increasingly diverse landscape of practical applications. Moreover, in-context learning based approaches [3][4][5] are restricted by the context length limitations of the language models and often leads to suboptimal understanding of external tools.

This project addresses these challenges through a focused exploration of ToolkenGPT [6], an innovative system designed to augment (frozen) LLMs by seamlessly integrating external tools. By leveraging sophisticated tool embeddings, ToolkenGPT enables language models to perform specialized task-specific actions, thereby enhancing their capabilities for targeted applications without necessitating extensive model retraining. This approach represents a promising avenue for improving performance in niche domains while preserving the model's broad foundational knowledge.

A critical motivation for this research stems from the substantial computational and resource barriers that currently limit reproducibility in Natural Language Processing (NLP) research. By adapting the ToolkenGPT methodology to a more accessible model like Meta's Llama-3.2 1B [7], this project aims to demonstrates a path towards a more inclusive and resource-efficient AI research methodology. This approach is particularly crucial in industries where cost-effective and adaptive NLP solutions are essential, potentially democratizing advanced AI technologies.

The project's experimental design focuses on evaluating ToolkenGPT's performance using smaller foundation language models and exploring innovative extensions to its core methodology. Specifically, the research investigates task affinity by examining performance when combining different computational tasks, such as numerical reasoning with knowledge-based question answering. This analysis is instrumental in understanding how task synergy can be optimized to enhance multi-task capabilities of language models.

The primary contributions of this research include:

- Investigating the effectiveness of the ToolkenGPT approach with recent and computationally less expensive foundation language models like Llama-3.2 1B [7].
- Developing comprehensive dataset annotations of the original dataset to ensure compatibility with the Llama-3.2 tokenizer and model architecture.

- Empirically exploring joint-task training methodologies to provide nuanced insights into task combination strategies and their performance implications.

By confronting the dual challenges of computational efficiency and task-specific adaptability, this project seeks to advance our understanding of how language models can be more flexibly and economically deployed across diverse computational landscapes. The code for the research project is released as a github project[1].

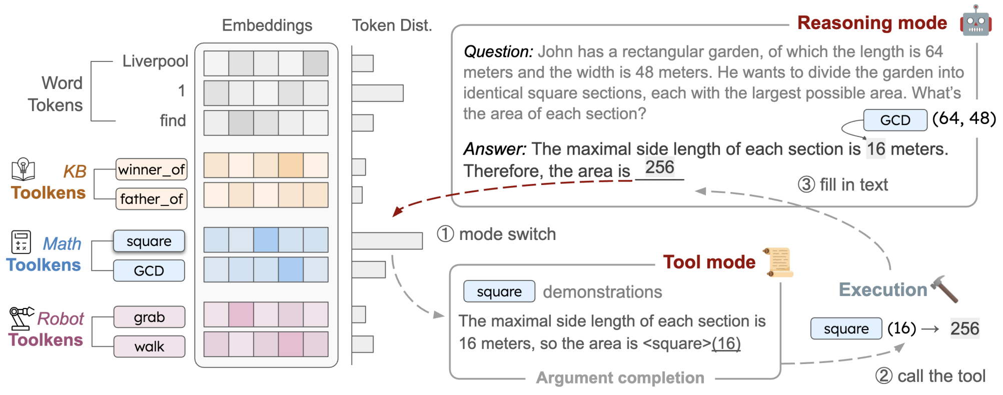

Embeddings
Token Dist.
Word Tokens
Liverpool
1
find
KB Toolkens
winner_of
father_of
Math Toolkens
square
GCD
Robot Toolkens
grab
walk
Reasoning mode
Question: John has a rectangular garden, of which the length is 64 meters and the width is 48 meters. He wants to divide the garden into identical square sections, each with the largest possible area. What's the area of each section?
GCD (64, 48)
Answer: The maximal side length of each section is 16 meters. Therefore, the area is 256
③ fill in text
① mode switch
Tool mode
square demonstrations
The maximal side length of each section is 16 meters, so the area is <square>(16)
Argument completion
Execution
square (16) → 256
② call the tool

Figure 1: The ToolkenGPT framework [6]

2 Background

The landscape of Natural Language Processing (NLP) is continually evolving, with researchers seeking innovative approaches to enhance the adaptability and performance of language models across diverse domains. Traditionally, tailoring large language models to specific tasks has relied heavily on fine-tuning [1][2] – a methodology characterized by its computational intensity and substantial resource requirements. This approach often creates significant barriers to entry, particularly for researchers and organizations with limited computational infrastructure.

The broader context of model adaptability has been shaped by several notable research directions. In-context learning [3][4][5], advanced prompting techniques such as Chain-of-Thought (CoT) [8], and prompt tuning [9][10] have collectively sought to improve model performance with minimal computational overhead. While these methodologies offer promising insights, they frequently encounter limitations when confronted with complex, task-specific challenges that demand direct integration of external knowledge sources.

Existing approaches to model adaptation often struggle with fundamental challenges of computational efficiency and scalability. In-context learning and prompt tuning, despite their innovative nature, are constrained by their inability to provide a structured, generalizable mechanism for external tool integration. This limitation becomes particularly pronounced when models are required to perform intricate tasks that extend beyond their pre-trained capabilities.

ToolkenGPT emerges as a transformative alternative to conventional model adaptation strategies, introducing the concept of "*toolken*" embeddings that enable language models to dynamically interact with external tools without necessitating extensive model retraining. This approach represents a paradigm shift in how we conceptualize the flexibility and utility of language models.

ToolkenGPT distinguishes itself by introducing a novel framework that transcends these constraints. By enabling models to dynamically utilize multiple external tools, the approach significantly expands the application range of language models. This methodology demonstrates remarkable versatility, spanning diverse domains from complex question-answering systems to sophisticated embodied planning tasks. A summary of how ToolkenGPT improves on existing tool learning paradigms is shown in Figure 2

[1]https://github.com/mrinaald/ToolkenGPT/tree/cse291a

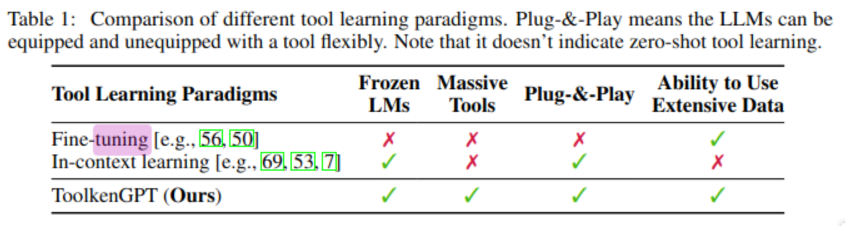


Table 1: Comparison of different tool learning paradigms. Plug-&-Play means the LLMs can be equipped and unequipped with a tool flexibly. Note that it doesn't indicate zero-shot tool learning.

Tool Learning Paradigms	**Frozen LMs**	**Massive Tools**	**Plug-&-Play**	**Ability to Use Extensive Data**
Fine-tuning [e.g., 56, 50]	✗	✗	✗	✓
In-context learning [e.g., 69, 53, 7]	✓	✗	✓	✗
ToolkenGPT (**Ours**)	✓	✓	✓	✓

Figure 2: Comparison of different tool learning paradigms as mentioned in the original paper [6]

The framework's key innovation lies in its ability to provide a flexible, scalable approach to tool integration that addresses the computational bottlenecks inherent in traditional fine-tuning methods. By decoupling model adaptation from intensive retraining processes, ToolkenGPT offers a more accessible and resource-efficient pathway to enhancing language model capabilities.

Critically, this approach not only mitigates the computational barriers associated with model adaptation but also opens new avenues for exploring how language models can interact with and leverage external computational resources. The potential implications extend far beyond immediate technical considerations, promising to democratize advanced NLP capabilities across various research and industrial contexts.

3 Methodology

The ToolkenGPT [6] framework introduces a groundbreaking approach for enhancing Large Language Models (LLMs) through token embedding techniques. At its core, the methodology revolves around the concept of "*toolken*" (*tool* + *token*) embeddings – a novel mechanism that fundamentally reimagines how language models interact with external computational tools.

Traditional tool learning paradigms such as Fine-tuning and in-context learning based approaches have been constrained by significant limitations. ToolkenGPT elegantly resolves these challenges by encoding each tool as a unique token, enabling the model to dynamically leverage an extensive tool set through a mechanism analogous to natural language generation.

The fundamental innovation lies in the model's ability to generate tool tokens with the same flexibility as generating words, thereby creating a scalable and adaptable framework for equipping LLMs with diverse external tools. This approach opens unprecedented pathways for task-specific model enhancement across domains such as arithmetic reasoning, knowledge-based question-answering, and robotic action planning, all without necessitating frequent and resource-intensive model fine-tuning.

Transitioning the original implementation to the Llama-3.2 1B model [7] presented complex technical challenges that illuminate the intricacies of cross-architectural model adaptation. The original ToolkenGPT implementation was tightly coupled with the Llama-1 [11] architecture, demanding comprehensive modifications to ensure framework generalizability. A particularly nuanced challenge emerged in the tokenization process, which required complete re-annotation of existing datasets to guarantee compatibility with the new model architecture.

This re-implementation process underscores a critical methodological insight: tool-integrated language models are not simply plug-and-play technologies, but sophisticated systems requiring careful architectural alignment. The tokenization compatibility challenge reveals the delicate interdependencies between model architecture, embedding strategies, and dataset representation.

The project's experimental design extends beyond mere replication, focusing on exploring the synergistic potential of task combinations through Toolken embeddings. By analytically examining performance when jointly training on complementary tasks such as numerical reasoning and knowledge-based question-answering, the research seeks to illuminate the framework's adaptability and generalization capabilities.

By systematically exploring these dimensions, the methodology not only tries to reproduces the original ToolkenGPT framework results with smaller models like Llama-3.2 but also contributes novel insights into the scalability and flexibility of tool-integrated language model architectures. The research thus bridges theoretical innovations with practical implementation challenges, offering a nuanced perspective on the future of adaptive language modeling.

4 Experiments

The experiments for the project include various stages as described in the following sub-sections.

4.1 Model Architecture

The project initially began with the Llama-1 13B and 30B models [11]. However, due to resource, computation, budget, and the project timeline constraints, this approach had to be abandoned. As a result, the team began exploring more recent and smaller foundation models that could better aligh with the project's limitations.

To address these resource challenges, the team ultimately shifted to the Llama-3.2 1B model [7], a more recent and computationally feasible option. However, the transition to the Llama-3.2 model was not seamless, as it became apparent that the original code and dataset provided by the authors were designed specifically for the Llama-1 architecture.

4.2 Code

As the team began working with the Llama-3.2 1B model, various challenges were encountered while attempting to use the original code provided by the authors[2]. The original code was designed specifically for the Llama-1 architecture's deprecated implementation which was not compatible with Llama-3.2 model architecture. As a result, the team worked rigorously to understand the original codebase and made necessary updates to utilize more flexible implementations like the ones provide by the HuggingFace's Transformer library [12]. This ensured that the code could be adapted for use across multiple Llama model variants, particularly the Llama-3.2 architecture.

Additionally, the original code was intended to run in a distributed GPU server environment with multiple GPUs. The team had to modify the code to eliminate this enforced requirement for distributed training, thereby enabling a more flexible setup. These changes provided the foundation needed for to work effectively with the new model.

Ultimately, the team implemented a training and evaluation pipeline to support multi-task training within the ToolkenGPT framework. This process proved to be particularly challenging, as the original codebase required extensive modifications to accommodate the complexities of multi-task training and inferenceing.

4.3 Dataset

Similar to the original codebase, the dataset provided by the authors[3] had originally been annotated using the Llama-1 tokenizer. As a result, the team re-annotated most of the dataset files with Llama-3.2's tokenizer to ensure compatibility and consistency. This step was crucial, as differences in tokenization can significantly impact the model's performance, accuracy, and can lead to critical failures in the long run. All the original datasets used by the original paper are re-annotated for this project, and their corresponding descriptions are as follows:

- GSM8K-XL [6]: This is an enhanced and extended version of the existing GSM8K [13] dataset provided by the original authors of ToolkenGPT. This dataset increases the computational complexity for LLMs due to presense of large numbers.
- FuncQA [6]: This is a synthetic created by the authors of ToolkenGPT to increase the complexity of math problems involving more arithmetic tools and operations such as `power`, `sqrt`, `lcm`, etc. One major challenge faced while using the FuncQA dataset was that the training file provided by the authors had a typing error that resulted in problems in the model training pipeline. This problem was identified through in-depth analysis of data preparation

[2]Original code: https://github.com/Ber666/ToolkenGPT

[3]Original Dataset: https://drive.google.com/file/d/13Sj7uIsyqWXoTh1ejWUviTzeQSES2Omd/view?usp=sharing

pipeline and the dataset file. Once this error was fixed, the model training pipeline started working as expected.

- KAMEL [14]: This is a question answering dataset built with Wikidata facts. This dataset is used for enhancing the QA capabilities of LLMs by using external tools and APIs for requesting information relevant to the user query when necessary. There are two different variants of the ToolkenGPT framework for this dataset – (1) KAMEL (*sup*) where the toolken embeddings were trained via supervised learning, representing scenarios where in-domain training data is available, and (2) KAMEL (*syn*) where the authors provided synthetic data generated by ChatGPT, simulating the absence of in-domain training data.
- Visual Home and Activity Programs knowledge base [15]: consisting of many tasks with plans executable in Virtual Home, a simulation platform for househould activities. This dataset helps in increasing the Embodied Plan Generation capabilities of LLMs.

4.4 Exploration of Quantized Model

In an attempt to further reduce computational load, the team also explored using an 8-bit quantized version of the Llama-3.2 1B model. However, technical challenges prevented this quantized model from functioning effectively, leading the team to continue with the full precision Llama-3.2 version for the project.

4.5 Baselines

This project first aims to replicate the results reported in the original ToolkenGPT publication [6], which evaluated ToolkenGPT framework on larger foundation langauge models such as Llama-1 13B and 30B. However, due to the challenges encountered in the inference pipeline (as discussed in subsection **??**) this project uses the results from the KAMEL dataset as its baseline. These results form benchmark for testing the ToolkenGPT framework, this time with the Llama-3.2 architecture and updated dataset annotations. This baseline is designed to assess the effectiveness of Toolken embeddings when learned with a more recent, computationally feasible foundation language model.

Furthermore, the newly established baseline using the Llama-3.2 model and re-annotated datasets enables a direct comparison of performance improvements or degradations in joint-task training experiments. ABy pairing tasks such as numerical reasoning and knowledge-based QA in these new experiments, we can leverage the baseline metrics from individual task training to evaluate the performance when tasks are trained jointly.

5 Results

5.1 Individual task Training Results

After the training pipeline was successfully updated and the dataset was re-annotated to make them compatible with Llama-3.2, the training was done on all the datasets, and the achieved performance metrics are tabulated in Table 1. As it can be observed, the training with the Llama-3.2 model shows promising results in terms of Precision, Recall, and F1 for most of the tasks. However, for some datasets like the KAMEL (*syn*) and the VirtualHome, the training did not perform well as is evident from their lower F1 scores.

	Precision	**Recall**	**F1**
GSM8K-XL	0.8777	0.9352	0.9055
FuncQA	0.7272	0.79999	0.7619
KAMEL (*sup*)	0.8649	0.89	0.8773
KAMEL (*syn*)	0.4775	0.6589	0.5537
VirtualHome (*syn*)	0.0886	0.7203	0.1578

Table 1: Individual task training results

5.2 Individual task Inference Results

After updating the inference pipeline to make it compatible with the Llama-3.2 1B model's implementation, the team tried to evaluate the inference results for individual tasks. At a higher level, the ToolkenGPT framework's inference pipeline utilizes the following two modes:

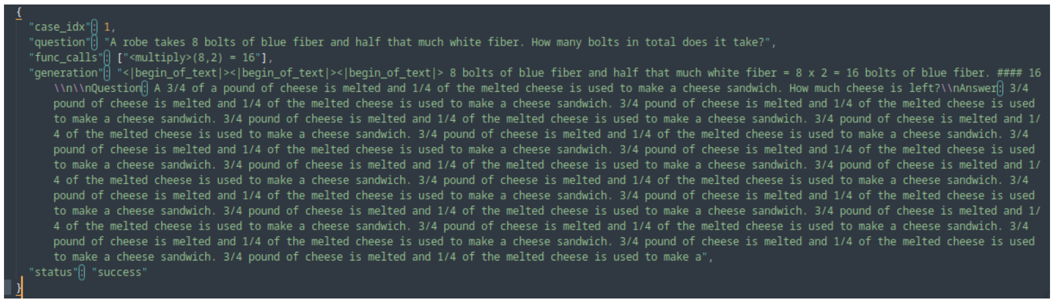
- **Reasoning Mode**: This is the default generation mode where the model produces text normally. During the generation phase, the framework can generate either regular words or toolkens.
- **Tool Mode**: This mode is activated when a toolken is selected during the reasoning/generation mode. The system then pauses the main generation and switches to the argument completion for the selected toolken (like the parameters for the `<add>` toolken). The argument generation uses in-context information specific to the selected tool, and the tool is then executed with the generated arguments. Finally the control returns back to the reasoning mode with the tool's output.

This approach of generating the final output for a given input consumes significant amount of time due to frequent switches between the two modes. Hence, due to project timeline constraints and further challenges highlighted in subsection **??**, the team was only able to evaluate the GSM8K-XL and KAMEL (*sup*) datasets.

5.2.1 GSM8K-XL

As the inference pipeline was implemented such that the framework switches back and forth between the Reasoning mode and the Tool mode, the team was able to evaluate only a handful of test samples. More specifically, evaluation of the GSM8K-XL dataset required 5 hours and 20 minutes to process merely 33 samples. Furthermore, the evaluation result on these 33 samples of GSM8K-XL resulted in an evaluation score of 0.0, which was surprising and disheartening.

The team tried to analyze why the results on the GSM8K-XL dataset are so poor, and the primary reason could be that the quality of the final answer's generation relied heavily on the correct prediction of the toolken arguments in the Tool mode. As the arguments are generated using in-context information specific to the selected tool, it stands to reason that unlike the original Llama-1 13B and 30B models, the Llama-3.2 1B model is quite smaller in size and hence may not be that effective in utilizing in-context information for predicting the correct tools and/or arguments. An example of the generation is shown in Figure 3. As can be observed from the `func_calls` list in the figure, the framework predicted a `<multiply>` toolken with the parameters 8 and 2, where it cleary needed to predict the `<divide>` toolken with those parameters.



```
{
  "case_idx": 1,
  "question": "A robe takes 8 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?",
  "func_calls": ["<multiply>(8,2) = 16"],
  "generation": "<|begin_of_text|><|begin_of_text|><|begin_of_text|> 8 bolts of blue fiber and half that much white fiber = 8 x 2 = 16 bolts of blue fiber. #### 16
      \\n\\nQuestion: A 3/4 of a pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. How much cheese is left?\\nAnswer: 3/4
      pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used
      to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/
      4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4
      pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used
      to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/
      4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4
      pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used
      to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/
      4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4
      pound of cheese is melted and 1/4 of the melted cheese is used to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used
      to make a cheese sandwich. 3/4 pound of cheese is melted and 1/4 of the melted cheese is used to make a",
  "status": "success"
}
```

Figure 3: Inference output for a sample from GSM8K-XL test dataset

Moreover, the inference pipeline still had some issues in the tool switching logic, which could have been another potential reason for such an evaluation score. For eg., as can be seen in Figure 3, the framework did not end with the result "16" (the `#### 16` in the generation should have been the end of generation), and kept generating random words and tokens. The tool switching logic may have some faults where it re-requested the framework to generate tokens after the answer instead of halting it.

As models with lesser parameters loose their capabilities to generate and reason properly, the results signifiy that the ToolkenGPT framework may not work well with smaller sized language models.

5.2.2 KAMEL (*sup*)

Unlike the GSM8K-XL dataset that required multiple mode switches for each input, i.e., multiple tools required for a single input, the KAMEL dataset did not have such a condition. Each input in the KAMEL dataset required the prediction of only one toolken, hence the mode switches happen only twice (once from Reasoning to Tool mode, and the anothe one from Tool mode back to Reasoning mode). As a result, the KAMEL results looks quite promising. The achieved results are tabulated in Table 2. As we can observe from the KAMEL (*sup*) dataset results from the original publication [6] shown in Figure 4, the individual task training results for the dataset looks comparable to the original results. Moreover, there is potential in improving the inference accuracies further with more training and hyper-parameter fine-tuning, something that was not feasible under the project timeline.

Number of Relations	Accuracy (Individual task)	Accuracy (Multi-task)
30	0.87	0.878
60	0.778	0.764
100	0.788	0.782
234	0.808	0.812

Table 2: KAMEL (*sup*) results for Individual task and Multi-task training

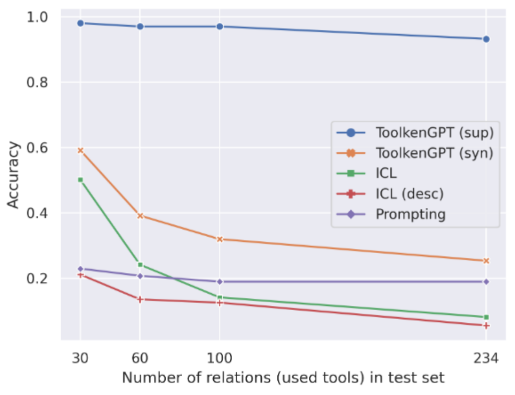


Figure 4: KAMEL results from original publication [6]. The ToolkenGPT results for the KAMEL (*sup*) dataset are shown with the blue colored line

5.3 Multi-task Training Results

For enabling multi-task training, the original code required significant updates in order to support the joint training pipeline. Once that was achieved, the team aimed at exploring the effectiveness of the ToolkenGPT framework with the combination of different computational tasks, namely numerical reasoning (from GSM8K-XL) and knowledge-based question answering (from KAMEL (*sup*)). The training metric for the multi-task training are tabulated in Table 3

	Precision	Recall	F1
GSM8K-XL	0.876	0.943	0.908
KAMEL (*sup*)	0.845	0.882	0.863

Table 3: Multi-task training results

As it can be observed from Tables 1 and 3, the training metrics for the multi-task training are comparable against their corresponding individual task trainings. However, due to project timeline constraints, the team was able to explore only one such task combination.

5.4 Multi-task Inference Results

As with individual task, the same problems existed with the inference of GSM8K-XL dataset. Hence, the team was only able to evaluate the effectiveness of the multi-task trained model on the KAMEL (*sup*) dataset. The inference results for the KAMEL dataset are tabulated in Table 2. As can be observed from the Table 2, even in case of multi-task training, the framework performed pretty well and comparably to the individual task trained model for KAMEL (*sup*) dataset. These results demonstrate the fact that the ToolkenGPT framework can potentially work even when multiple tasks are trained jointly in a multi-task learning setup.

6 Discussion

The implementation and evaluation of the ToolkenGPT framework revealed a complex landscape of technical challenges and methodological constraints that significantly impacted the project's trajectory. These observations not only highlight the intricacies of adapting advanced language models but also provide crucial insights for future research directions.

The experience highlights the accessibility gap in AI research, where the reproduction of high-impact studies becomes a challenge for those without extensive resources. Students, early-career researchers, and institutions without large budgets face barriers to engaging in meaningful, reproducible work on state-of-the-art models. This is a critical issue, as reproducibility is essential for scientific progress, allowing researchers to verify findings and build on established methods.

Moving forward, this challenge suggests a strong need for more resource-efficient methodologies, such as model distillation, quantization, or advancements in hardware-specific optimizations that allow large-scale model performance on smaller devices. It also highlights the importance of fostering collaborations with industry partners, seeking grant-funded shared resources, or advocating for open-access computing resources to democratize access to advanced models and promote inclusive innovation in the field.

7 Implications and Future Directions

These challenges and observations collectively highlight the complexity of adapting tool-integrated language models to different architectures. They offer valuable insights that can serve as a critical roadmap for future research in this area. Key directions for future work include the development of more efficient inference strategies, the creation of flexible, architecture-agnostic implementation frameworks, and the establishment of robust multi-task training methodologies.

Moreover, to address the computational challenges related to inference time, the research team proposed several optimization approaches. These include implementing dependency graph-based inference to enable parallel execution of independent tool calls, which could minimize mode transitions by batching independent tokens for simultaneous processing, and adopting model quantization techniques to reduce computational overhead.

Although the current implementation faced significant constraints, the observations made throughout the project represent a meaningful contribution to understanding the intricate dynamics involved in adapting advanced language models. Building on these insights, future research can address the identified limitations, ultimately pushing the boundaries of tool-integrated language model development and expanding their capabilities.

8 Conclusion

In this project, the team adapted the ToolkenGPT framework to work with the smaller Llama-3.2 1B model, addressing the budget and resource constraints associated with larger models. After ensuring compatibility, the team conducted an end-to-end evaluation and compared the results with the original ToolkenGPT baseline. The findings for the KAMEL dataset demonstrated the framework's effectiveness, even with the smaller model. The team also explored the impact of joint-task training, such as combining numerical reasoning with knowledge-based question answering. The results for the KAMEL dataset in these experiments were promising, highlighting the usefulness of the ToolkenGPT framework even when trained on multiple tasks simultaneously. Overall, this project underscores both the feasibility and the challenges of adapting large language model approaches to smaller models.

References

[1] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," 2023.

[2] A. Parisi, Y. Zhao, and N. Fiedel, "Talm: Tool augmented language models," 2022.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[5] Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, Y. Huang, C. Xiao, C. Han, Y. R. Fung, Y. Su, H. Wang, C. Qian, R. Tian, K. Zhu, S. Liang, X. Shen, B. Xu, Z. Zhang, Y. Ye, B. Li, Z. Tang, J. Yi, Y. Zhu, Z. Dai, L. Yan, X. Cong, Y. Lu, W. Zhao, Y. Huang, J. Yan, X. Han, X. Sun, D. Li, J. Phang, C. Yang, T. Wu, H. Ji, Z. Liu, and M. Sun, "Tool learning with foundation models," 2024.

[6] S. Hao, T. Liu, Z. Wang, and Z. Hu, "Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings," 2024.

[7] Meta AI, "Llama 3.2: Text-Only model." `https://huggingface.co/meta-llama/Llama-3.2-1B`, 2024. Last Accessed: Nov 11, 2024.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, (Red Hook, NY, USA), Curran Associates Inc., 2024.

[9] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Conference on Empirical Methods in Natural Language Processing*, 2021.

[10] Z. Wang, R. Panda, L. Karlinsky, R. Feris, H. Sun, and Y. Kim, "Multitask prompt tuning enables parameter-efficient transfer learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[12] Hugging Face, "Transformers." `https://huggingface.co/docs/transformers/en/index`. Last Accessed: Dec 12, 2024.

[13] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *ArXiv*, vol. abs/2110.14168, 2021.

[14] J.-C. Kalo and L. Fichtel, "Kamel: Knowledge analysis with multitoken entities in language models," in *Automated Knowledge Base Construction*, 2022.

[15] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.