AI Math Aid

Peter Dunbar
CS 4820-001, 1420 Austin Bluffs Pkwy, UCCS, CO 80918 USA

Abstract

By tuning the TinyLlama model, we will be able to quickly and accurately understand unique questions from user input using the process of tokenization and lemmatization, as well as the use of LoRA. The model will predict and find the best matching intent, then begin generating a response that fits the need of the user. Using a dataset that contains both questions and stepped answers will allow it to cover a wide range of math questions. This paper also covers basic concepts of how NLP models work and how dataset looks using intent-based Design.

Introduction & Significance

The use of online resources in modern day education has increased dramatically. Online applications like "MathWay", "Photomath", "SnapCalc" are being used to answer problems immediately. While these applications are fast and accurate, the student misses out on how to step through the problem and is unable to ask clarifying questions to the application. This ultimately leads the student to failure, if not immediately then certainly down the line in their education. Math acts as a different language, so understanding it as well as how to speak math is also important in a model's responses. If each response can fully explain a problem and how to solve it step by step, then a student can get a full image of the math problem and better understand it.

A Chatbot that is able to respond to math questions while guiding the user step by step would deliver both the speed of the application but also deliver the understanding that a tutor can provide. Our model plans to include an Intent based design to better understand what the user is trying to ask. Following DNN / Feedforward our model will be able to understand user input. The Natural Language Processing (NLP) models will utilize both tokenization and lemmatization to easily understand typed math language. Our training will include methods to address overfitting to avoid degradation of the model over time.

Furthermore, mathematical expressions for intelligent tutoring systems are distinctly challenging because, unlike language, they are symbolic, context-dependent, and rely on vertical and horizontal arrangement. For instance, the symbol "x" could be a multiplication sign or a variable. It is also important to note that expressions like fractions, exponents, and integrals are spaced vertically, not horizontally, unlike language. An effective AI tutor must be able to solve problems and recognize the mathematical symbols correctly within the context of a problem.

Currently, the ability of neural networks and large language models (LLMs) to handle symbolic computation remains limited. (Lample and Charleton 2020). This lack of ability is mainly because mathematical expressions need to be parsed structurally and linked to their definition instead of being interpreted sequentially like human language (Lynch, Eck, and Ploennigs 2024). Standard tutoring systems are limited to specific problem types and struggle with explaining step-by-step symbolic computation due to this limitation. Often, these tutoring systems will gloss over or entirely miss important computational steps. Given these points, neural networks have been widely used for symbol recognition and computation, showing great promise. However, their accuracy declines when applied to noisy or incomplete student work (Kukreja and Sakshi 2022).

This project aims to create an AI tutor that can accept any form of a K-12 math problem and solve it successfully. It must also be able to correctly interpret typed student solutions to give the most helpful feedback. Correctly parsing student work is important since people learn best from working out problem solutions rather than problem solving alone (Barbieri et al., 2023). Thus, a model that accurately recognizes typed symbols, parses them into structured representations, and evaluates their meaning must be designed to achieve this. Any misunderstanding of symbols from the AI tutor can halt a student's ability to learn. Hence, this task is crucial because it results in a tutor that can identify mathematical mistakes and explain concepts as well as solve complex problems.

Intent-based Design

Intent-based design means that our data set will be modeled after certain intents that define the purpose of the data.

This is a file that contains all of the potential scenarios the model is expected to encounter. Common intents for this model will include all the core subjects, fundamentals, and basic greetings. Each intent has "a tag field defines the intention of that particular intent" (Vamsi, Rasool, and Hajela 2020). These tags allow the model to organize data underneath the intent umbrella.

Each "A tag is a keyword assigned to the patterns and the responses for training the chatbot" (Vamsi, Rasool, and Hajela 2020). A set of tags for an intent like "explainArithmetic" would include a set of tags covering addition, subtraction, multiplication, division, fractions, decimals. Each tag then has another layer underneath it being patterns.

The patterns "are the types of queries asked by the user to the chatbot" (Vamsi, Rasool, and Hajela 2020). For this model the patterns will primarily be the most common way someone will ask a question see figure 1 for an example. These patterns aid the model in parsing data faster by aiding in its prediction of where certain information is stored. Sharing the same umbrella as patterns is responses, these "Responses are the answers generated by the chatbot for the respective queries" (Vamsi, Rasool, and Hajela 2020). This model will include some generic responses to ensure a consistent speech pattern across the different subjects, see figure 1 for an example.

"Fallback responses are needed for any question that does not fit the model's intents" (Vamsi, Rasool, and Hajela 2020). This includes questions about other subjects, questions about the model itself, and really anything non-math related. This will be kept simple with one response needing to be listed. Overall, this model will include around forty total intents in order to cover the K-12 curriculum.

```
{"tag": "explainMultiplication",
 "patterns": ["How do I multiply",
              "explain how I can multiply",
              "Show me how to multiply"],
 "responses": ["Let's figure this out together! Do you want to see an example of
                your problem or something similar?"]
},
```

Figure 1. Intent example

Tokenization & Lemmatization

Tokenization is "converting the character stream into a set of individual tokens. Tokens can be words, numbers, identifiers, special characters, or punctuation" (Bhartiya et al. 2019). This is taking a user input and breaking the task down into simpler pieces and removing anything that isn't needed for understanding the input. This model will be able to break down the math vocabulary. A user could write out in English what their math question is. For example: "What is two plus two to the square root of two all over two". This model will be able to break this sentence down and arrive at "(2 + 2^√2)) / 2", by creating tokens out of words like "square root" and "plus". Tokenization will be extremely important for the models understanding as every symbol has an English word.

Lemmatization is "reducing word variations into simpler forms. Lemmatization uses a language dictionary to preform an accurate reduction to root words" (Bhartiya et al. 2019). This action takes place after Tokenization. The word multiply could be said as "multiplies", "multiplying", "multiplied" and in many different contexts but it always means multiply. Our model will be able to understand more unique questions faster like "after six has been multiplied by five, multiply it by two". This model will break it down into "six has been multiply by five multiply it by two". The last issue is the unnecessary words between.

Stop word removal is a practice that discards words like "'is', 'are', 'the', etc. do not add value to the meaning of the sentence. They are not as important as the keywords, hence they can be removed from the text for better processing" (Bhartiya et al. 2019). Looking at the example from the last paragraph our result would end up looking like "six multiply five multiply two". Now the user's question is easily understood, and the model can now start generating a response in far less time.

DNN model

Showing the steps of a problem is not the same as explaining the steps of a problem. In order for this model to be able to explain these user questions fully a Deep Neural Network is required. DNN is a "neural network with multiple layers is bult for processing the input data or the training data so that the model can extract higher-level features of the data" (Bhartiya et al. 2019). This model will be "trained on large amounts of data/information to learn the procedure of responding with relevant and grammatically accurate responses to input utterances" (Vamsi, Rasool, and Hajela 2020).

The architecture of this model will follow a feed-forward neural network type. This means that the information of the system will only move forward in the network. This flow will start at the input layer then go to the middle transforming layer which includes tokenization and lemmatization, and finally the output layer which is mapped by the intent classes in the model.

A potential issue in this model is overfitting which is "When a model learns the detail and noise in the training data to an extent that it negatively impacts the performance if the model on the new data" (Bhartiya et al. 2019). Early stopping and the Dropout technique can be used to prevent this.

Early stopping "is used to stop the training at the point when performance in the dataset starts to degrade" (Bhartiya et al. 2019). This is a quick and simple way this model can prevent itself from learning from noise.

The dropout technique “is a simple and powerful regularization technique where randomly selected neurons are ignored during training” (Bhartiya et al. 2019). This makes the model choose new paths within its network preventing potential over reliance on a certain path

Response Types

This model in order to improve the vocabulary, will include response types. These types will add a more human dialogue to the model to better help the user understand and process responses from the model and stay on topic. For example if the user were just getting the same bland response over and over would make the user natural drift from the task at hand.

An important type is sequential responses. This means that the model after multiple inputs start “recognizing the pattern of similar questions being asked repeatedly, varied responses were added for frequently recognized intents” (Colace et al. 2018). These changes could be as small as verbiage or just the context.

The Aha response is a variant that will, after a concept has been understood or an error has been corrected by the user, say something like “Oh, I got it! Now that I put all shaded pieces together, I think the total amount should be greater than one cup because both the red and blue parts are more than half. Thanks much!” (Colace et al. 2018). This helps to pacify any potential annoyance with the model.

These response types can also cover other tones and contexts. A potential response type for math tutorage could be a define response where the response not only comes with an answer but an subsequent definition of any terms or symbols in the users input.

Tiny Llama

This project will be running off of the model TinyLlama, “a compact 1.1B language model pretrained on around 1 trillion tokens for up to 3 epochs1" (Zhang, P et al., 2024). It is an open-source language model that will be behind the math problem reasoning and the communication with the user. TinyLlama was made to be smaller, which is perfect in this case as this project will most likely be running on devices with less computational power. Even tough this model is small it is still able to handle larger dataset and can still preform well. TinyLlama was created “Following the same architecture and tokenizer as Llama 2”(Zhang, P et al., 2024). TinyLlama also has a specialized model that can handle both math and code better. This could be another option to further optimize the creation of the MathAid. This option will be explored as to allow for something to compare the normal TinyLlama against.

MATH dataset

This dataset is important to the training of MathAid. This dataset is highly used are regarded as crucial in training models for understanding math questions and its solving ability. The MATH dataset is made up “of 12,500 challenging competition mathematics problems”(Hendrycks et al., 2021). The dataset also contains ways to assess the models ability to problem-solve.

An amazing part of this dataset is that “Each problem in MATH has a full step-by-step solution which can be used to teach models to generate answer derivations and explanations”(Hendrycks et al., 2021). This is super important for the models ability to analyze a problem to be able to explain it. Another part is that each problem is ranked into difficulties which will allow for checking performance.

Tuning

Part of tuning is editing or using a dataset that will provide not just answers and solutions but also includes step-by-step instructions, as "In contrast, having models train on solutions increases relative accuracy by 10% compared to training on the questions and answers directly"(Ding et al., 2023). Using a dataset like MATH will be more advantageous do to the model being designed for not only answering but explaining questions.

Goals for next section

Our goals with this model is to tune TinyLlama to be-come a chatbot capable of both explaining math and speaking math. The model will sound more human to better help the user understand. Some milestones in this will include the first testing using the MATH dataset, achieve a higher confidence, implementing various tech-niques, and including varied responses to increase per-sonification.

Testing & Creation

As explained in the previous section of this paper the over all process is as follows: We take the TinyLlama model and the MATH-500 dataset from hugging face then wrap TinyLlama in LoRA using PEFT, this allows us to use the model more efficiently and better tune the hyperparameters. After the parameters are set we train the model on a 400-100 split and then send the latest checkpoint of the model to prompting, for the examples shown in this paper the generic question will be ‘What is 2 + 3?’. The following subsections will go over my findings and constraints with the resources we’ve chosen to use.

Using LoRA

With LoRA we could potentially use larger LLMs like ChatGPT or Llama. Beyond that it is still very useful for training TinyLlama as loRA “is a method that reduces memory requirements by using a small set of trainable parameters, often termed adapters, while not updating the full

model parameters which remain fixed"(Dettmers et al., 2023).

A potential path forward for this project is using a higher grade LLM like Llama and then using QLoRA as in the paper 'Efficient Finetuning of Quantized LLMs' their "results show that 4-bit QLORA is effective and can produce state-of-the-art chatbots that rival ChatGPT"(Dettmers et al., 2023). So, if our model is unable to perform well enough we can possibly switch to more in depth methods using QLoRA and a Llama to boost to overall 'power' of the MathAI model.

The parameters used for the training and subsequent results is as seen in figure two. The most notable parameters are epochs, and the learning rate. During early testing I had set both to low and ended up getting results that weren't even remotely answers. After this I increased the epoch steadily up by 1 each training and eventually landed on 5, which in total is 800 checkpoints. Around the 750-780 checkpoint mark the train loss dips below 0. The training speed was increased and due to the results I assumed that the dataset was the largest problem, but something that was pointed out to us was that the training speed could be running to fast that the model was being trained on noise at some point, we hope to find a good speed for the next iteration.

```
training_args = TrainingArguments(
    output_dir="./tinyllama-math",
    eval_strategy="steps",
    save_strategy="steps",
    save_steps=100,
    logging_steps=10,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    num_train_epochs=4,
    learning_rate=5e-4,
    fp16=True,
    report_to="none", #wandB
)
```

Figure 2. Training parameters

Using MATH dataset

The MATH-500 dataset is well made and has really good examples but the overall size of the set is hindering the results of the model during training. They mention that while "Instruction Tuning (Wei et al., 2021) is an effective approach to unlock certain capabilities in LLMs. Unfortunately, this approach is constrained by the limited size of the currently available datasets on mathematical reasoning" (Tang et al., 2024). Meaning that current datasets being used for training are just not large enough to fully train a LLM meaningfully. So, we intend to use MathScaleQA a "mathematical reasoning dataset (MathScaleQA) containing two million math question-answer pairs"(Tang et al., 2024). This dataset was applied "to fine-tune open-source LLMs (e.g., LLaMA-2 and Mistral), resulting in significantly improved capabilities in mathematical reasoning." (Tang et al., 2024). With this change of datasets, we hope to improve the overall results. We intend to test the parameters listed in the paper used or Llama, for the MathAid model.

TinyLlama Math & Code model

The TinyLlama math & code model did extremely poorly as we believe this is due to the nature of the dataset and the task we are trying to achieve. The math & code model is built more for just raw inputs of math questions and code. So when we trained it to try and explain the steps it was going through it would just return noise, but the off thing about this noise is that it normally had nothing to do with the original question, see figure three.

The even more concerning part was that it was also unable to provide a correct answer to the problem given to it while the chat model could at least get that far. Both models were using the same LoRA wrapper parameters, see figure four. They also shared the same training parameters as seen before in figure two.

Overall, this model only takes less than thirty minutes on average to fully train, but the avg training loss is around '2', see figure five, which is incredibly bad, especially in comparison to the chat model, it is very unlikely that we will be returning to this model.

```
Device set to use cuda:0
What is 2 + 3?
Answer the question step by step:</s>
<|assistant|>
Step 1: Identify the main subject and action in the sentence.
The main subject is "the man," and the main action is "hitting the ball."

Step 2: Determine the location of the action.
The action takes place at a "tennis court."

Step 3: Determine the context of the action.
The context of the action is "hitting the ball."

Step 4: Determine the person or thing involved in the action.
The person or thing involved is "the man," as he is the one hitting the ball.

Step 5: Determine the purpose of the action.
The purpose of the action is "hitting the ball."

Step 6: Formulate a sentence that reflects the information from the given sentence.
The man hits the ball.</s>
<|assistant|>
```

Figure 3. Math & Code model results

```
config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
)
model = get_peft_model(model, config)
```

Figure 4. LoRA configuration

700	2.324500	2.686564
710	2.206100	2.686564
720	2.755700	2.686564
730	2.607100	2.686564
740	2.094800	2.686564
750	2.404800	2.686564
760	2.229600	2.686564
770	2.800900	2.686564
780	2.818400	2.686564
790	2.892400	2.686582
800	2.124000	2.686555

Figure 5. Math & Code Training loss

TinyLlama Chat model

The TinyLlama chat model has way better results then the math & code. We believe this is due to the nature of the data set and the task we are trying to achieve. The chat model has an easier time with the dataset as the set includes a lot of text describing how to step through problems alongside the actual solving of said math problems.

The results, as seen in figure six, show us that the overall training loss is at a good spot after four epochs. But as seen in figure five the steps for the question are just noise. Some solutions for this were brought up to us for this issue: The learning rate could be potential too fast, see figure two. We could use a larger dataset or simply double the current one. Lastly, we could include 'PEMDAS' in the prompt to give the model a baseline method for analysis of problems given to it.

Overall, this model only takes less then forty minutes on average to fully train, and avg results on training loss tend to be below '1', see figure seven, this tells me that the model itself is doing great but like as mentioned above the dataset is to small and the learning rate may be too fast.

```
Device set to use cuda:0
What is 2 + 3?
Answer the question step by step:  [t]
2 + 3 is 5.
Answer step by step:
2 + 3 is 5.
Answer step by step:
How many $\times$ 5 is 24?
Answer step by step:
How many $\times$ 5 is 24?
Answer step by step:
How many $\times$ 5 is 24?
Answer step by step:
How many $\times$ 5 is 24?
```

Figure 6. Chat model results

700	0.931900	0.962656
710	0.891800	0.959331
720	0.946100	0.956888
730	0.815400	0.954551
740	0.808200	0.952996
750	0.928900	0.951351
760	1.002200	0.949497
770	1.031900	0.948341
780	0.910900	0.947345
790	0.980900	0.946579
800	0.855500	0.946144

Figure 7. Chat Training loss

Cumulative Reasoning (CR)

This is a concept we came across to late into the testing so its use and results will be described in the next section of this paper, for now we'll go over how it can potentially improve results and better help the creation of the 'explanation' steps.

"CR decomposes a problem into manageable sub-tasks and incrementally builds a solution by cumulatively accumulating and verifying intermediate reasoning steps" (Zhang et al., 2025). So in just about every math class the teacher will explain a problem one step at a time. They will go over the step and why the previous step led to the current one. Afterwards they will use the context of the current step to create an answer for the next step, this is the general idea of CR.

In terms of prompting with LLMs its normally a good idea to break your queries into multiple steps for the model to step trough, in our case we'll be asking it to only go down one step at a time, then we ask it to go on to the next step using the previous ones as context.

This functionally recreates how we as humans would think to step trough a problem. This will hopefully not only improve the quality of steps but also improve the readability and understanding for the user.

Inside the paper they got the subsequent results: "In solving MATH problems, CR achieves a 4.2% increase from previous methods and a 43% relative improvement in the most challenging level 5 problems" (Zhang et al., 2025). So this is not a complete fix to the problem that we are having with the noise but it should notably improve results once implemented alongside an increase dataset and finer tuning of the parameters.

Reflections & Goals for next section

Overall the initial experiments were able to at least produce results but unfortunately none of them were anything to impressive. It did however point out that a potential dataset change alongside a slowing of the learning rate could significantly improve results. This on top of implementing CR will hopefully get the model where we want it to be for the final. The model that will be used for future testing will most likely be the TinyLlama chat model as on average its able to out preform the Math & Code model due to the type of dataset we are training it off of.

References

Chatbot for E-Learning: A Case of Study
Colace, F.; De Santo, M.; Lombardi, M.; Pascale, F.; Pietrosanto, A.; and Lemma, S. 2018. Chatbot for E-Learning: A Case of Study. *International Journal of Mechanical Engineering & Robotics Research* 7(5): 528-533

Efficient Finetuning of Quantized LLMs
Dettmers, Tim; Pagnoni, Artidoro; Holtzman, Ari; Zettlemoyer, Luke. 2023. *QLoRA: Efficient Finetuning of Quantized LLMs*. In *Advances in Neural Information Processing Systems* (Vol. 36), Curran Associates, Inc., pp. 10088–10115. https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf

Parameter-efficient fine-tuning of large-scale pre-trained language models
Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.-T., Chen, J., Liu, Y., Tang, J., Li, J., & Sun, M. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature MachineIntelligence,5*(3),220–235. https://doi.org/10.1038/s42256-023-00626-4

Chatbot: A Deep Neural Network Based Human to Machine Conversation Model
G. K. Vamsi, A. Rasool and G. Hajela, "Chatbot: A Deep Neural Network Based Human to Machine Conversation Model," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225395.

Measuring Mathematical Problem Solving With the MATH Dataset
Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. *Measuring Mathematical Problem Solving With the MATH Dataset*. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks. https://doi.org/10.48550/arXiv.2103.03874*

Artificial Neural Network Based University Chatbot System
N. Bhartiya, N. Jangid, S. Jannu, P. Shukla and R. Chapaneri, "Artificial Neural Network Based University Chatbot System," 2019 IEEE Bombay Section Signature Conference (IBSSC), Mumbai, India, 2019, pp. 1-6, doi: 10.1109/IBSSC47189.2019.8973095.

MathScale: Scaling Instruction Tuning for Mathematical Reasoning
Tang, Zhengyang, Zhang, Xingxing, Wang, Benyou, and Wei, Furu. *MathScale: Scaling Instruction Tuning for Mathematical Reasoning.* In Proceedings of the 41st International Conference on Machine Learning (ICML 2024), edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Proceedings of Machine Learning Research, Vol. 235. JMLR.org, 2024. https://dl.acm.org/doi/10.5555/3692070.3694024

Cumulative Reasoning with Large Language Models
Zhang, Yifan, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. *Cumulative Reasoning with Large Language Models.* Transactions on Machine Learning Research (TMLR), Journal of Machine Learning Research (JMLR), 2025. https://openreview.net/forum?id=grW15p4eq2

TinyLlama: An open-source small language model
Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). TinyLlama: An open-source small language model. *arXiv preprintarXiv:2401.02385*. https://arxiv.org/abs/2401.02385