Project: AI Tutor For Students

Team Members:

1. Anish Shrestha
2. Rachana Subedi
3. Shashwot Shrestha

LITERATURE REVIEW DOCUMENT :

AI_Tutor_for_Students(Literature Review)

https://docs.google.com/document/d/1mrLiHYKq7CocyhQgVdGxGXapxMsLLK6eIdO3S4vwA2s/edit?usp=sharing

Literature Review

Introduction

Our project is divided into three major parts. They are: Mathematical Problem Solving, Code Debugging Assistance, and Conceptual AI Question-Answering. All of these components are to be implemented with a chat bot.

The main research question of our project is, “ How can we utilize LLMs to help students in three major tasks: solving mathematical problems, answering AI related questions, and debugging the code ?”. The most suitable way to accomplish the project is by leveraging the power of LLMs and transformer based models.

The main flow of the project can be divided as follows:

- For all parts of the project, finding the most suitable transformer based models respectively.
- Collecting the necessary datasets for each part.
- Fine Tuning the model on the datasets we collect.
- Using RAG and prompt engineering to enhance the performance of the model.

General Review

We did literature review considering three main parts of our project as standalone components. Most of the previous work for all the three parts of our project is done with LLMs mostly by:

i) Prompting frozen LLMs,
ii) Utilizing strategies enhancing frozen LLMs, and
iii) Fine-tuning LLMs.

For mathematical problem solving, the general idea is to amplify the reasoning ability of LLM in solving mathematical problems. There is a lot of research with Chain-of-thought and Program-of-thought approaches, self feedback methods, and pretraining and filetuning of LLMs. The most popular datasets related to mathematical domains include MATH and GSM8k. Also there are numerous datasets available for this and hence enough datasets for developing mathematical models, just the constraint is compute resources for now. LLEMMA and ChatGLM-Math, both seem to be promising mathematical models. But it is to be noted that most work on ChatGLM-Math is done in Chinese.

As for the code debugging, the convenient way is to use external tools to run the code and extract error messages. Combining both buggy code and error messages, with suitable prompt engineering, asking LLM for a fix can help debug the code. Any proper dataset used for code debugging tasks are not open-sourced anywhere and prompt engineering is the way to go. Some popular LLMs for code completion tasks are Starcoder, deepseek coder, and Copilot. Also, when the dataset is not enough for finetuning like in our case of AI Q&A, the usual approach is using RAG along with prompt engineering to guide LLM in the proper path for generating a suitable answer. Most of the LLMs seem good for general question answering tasks, even for tasks related to conceptual AI questions. However, there is no specific benchmark which illustrates which model is better for answering AI related conceptual questions only.

Paper Review

1) Title: Can we learn from developer mistakes? Learning to localise and repair real bugs from real bug fixes.

Authors: **Cedric Richter**(University of Oldenburg Oldenburg, Germany), **Heike Wehrheim**(University of Oldenburg Oldenburg, Germany)

DOI: https://arxiv.org/abs/2207.00301v1

The main idea of this paper is to develop a code debugger model (RealiT), which is pre-trained on a large number of artificial bugs produced by traditional mutation operation and fine tuned on a smaller set of real bug fixes. The research question that is targeted by the authors is “ Can a model pre-trained on code mutants and fine-tuned on real bug fixes localise and repair more code bugs than traditional models like RNN, GNN, and GREAT ”. The authors found that pre-training on mutants and fine-tuning on real bug fixes combines the strength of both the high localisation and repair performance and the bug detection accuracy.

Main contributions

- A novel pre-train-and-fine-tune approach for learning to localise and repair bugs with Transformers is developed.
- The authors have shown that pre-training on mutants plays an important role for achieving high performance level (but pre-training only does not unlock the full power of the model) and fine-tuning on real code bug fixes made the model better than current state of art models.

Datasets

For training and evaluating the model two types of datasets are used. For the code mutants a general Github corpus of code snippets is used. As a general corpus of python code, ETH Py150k dataset has been used. This dataset contains 150k program files from popular python projects. The dataset is split into 100k files for training and 50k files for testing.

For the real code bug fixes SSB-9M dataset is used. It consists of over 9M general single statement bug fixes in Python. The dataset does not include the necessary implementation code itself but references the original commits in the repositories in addition to other useful metadata.

This project does not use any pretrained models, rather it builds the model for scratch.

Procedure

First step is the training and fine tuning step.

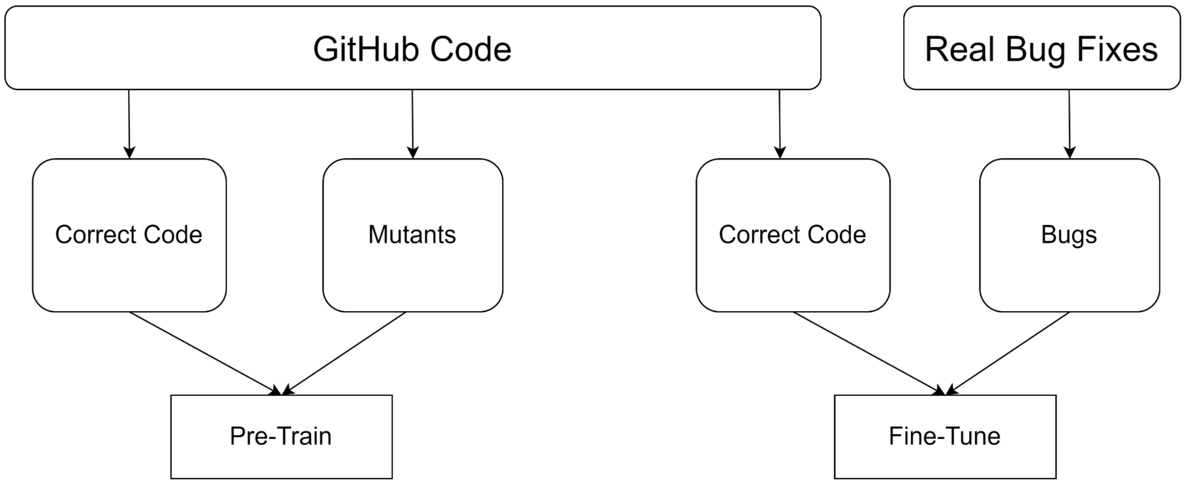


Fig 1: Pre-training and fine-tuning

Next step is evaluating where the model tries to find the bugs in the code and localise and repair the bug.

Role of Transformer

The source code is represented by a sequence of tokens T = *t*0, *t*1, *t*2, . . . , *t*n. Each token represents a possible bug location. Single token bugs are fixed only by replacing single tokens *tl* with another token r. Each token is embedded by employing BPE subtokens. At last, the sequence of token embedding is encoded via a transformer encoder to obtain a contextual vector e(t) for each token.

To finally compute the probability distribution over bug locations and repairs, the contextual vector of each token is used by localisation and repair modules. The localization module is a multilayer perceptron that computes a bugginess score for each potential bug location based on the vector representation e(t) and the original token embedding. The objective of the localization module is to learn how likely the token *ti* does not fit its surrounding context. The repair module is designed similar to CopyNet. Given the vector representation e(t) of a potential bug location, the repair module computes a repairing score between the bug location and each repair candidate at token *tj*.

So, with the help of each context vector e(t) the transformer module can find the bug location by analysing the context of each token, and also find the best replacement token from the vocabulary of the code.

One thing to be noted is that this project can only work for a single token bug in python.

Performance Metrics

1. Bug Localization and Repair Accuracy

    - **Localization Accuracy**: This measures how well the models can pinpoint the location of a bug in the code.
    - **Repair Accuracy**: This assesses the models' ability to generate the correct fix for the identified bug.

    The performance was evaluated before and after fine-tuning on real bug fixes. The project found that fine-tuning significantly improved both localization and repair accuracy. Interestingly, the performance gain for localization was higher than for repair, especially as the number of mutants increased up to 100x.

2. Comparison of Models

    - The performance of the RealiT model was better compared with other models such as RNN, Transformer, GNN, and GREAT.

- RealiT showed significant improvements in localization and repair accuracy compared to these baseline models, particularly when fine-tuned on real bug fixes.

3. False Positive Rate (FPR)

- The FPR was measured on bug-free code snippets to evaluate how often the models incorrectly identified bugs.

- RealiT had a comparable FPR to other models, with a slight increase in false positives considered acceptable given its superior bug localization and repair capabilities.

Key Learning And Finding from the Paper

I learned how the transformer model can be used to find bugs in the source code by using the information of the context vectors. Also, this paper gave a clear explanation of how the pretraining and fine tuning both can enhance the power of LLM. It gave a complete explanation about the process of localising and repairing bugs from scratch. Also, it gave insight about generating buggy code with the help of code mutation.
The most important finding is that, only pre-training on code mutants does not give a good accuracy model because the model is trained only on code mutants, which may differ from the actual mistake made by human programmers. So, fine-tuning on real code bug fixes is a must. The reason behind using mutants to pre-train the model is because of the low number of real code bug fixes data available publicly and the mutant enhances the power of the model since most of the mutants matched with the real bug made by the programmers.

2) Title: Repair Is Nearly Generation: Multilingual Program Repair with LLMs

Authors: **Harshit Joshi**(Microsoft, India), **Jose Cambronero Sanchez** (Microsoft, USA), **Sumit Gulwani**(Microsoft, USA), **Vu Le**(Microsoft, USA), **Ivan Radicek**(Microsoft, Croatia), **Gust Verbruggen**(Microsoft, Belgium)

DOI: https://doi.org/10.1609/aaai.v37i4.25642

In this paper a multilingual code repair engine (RING) is developed. This model leverages the power of LLMs by using a large language model trained on code (LLMC) such as Codex. This model enables flipped model for programming assistance, one where the programmer writes code and the AI assistance suggests fixes. Here, prompt engineering is used in order to localisation and transformation of the bugs. The authors have evaluated the model on 6 different languages and compared performance to language-specific repair engines. They have shown that RING can outperform langua-gespecifc repair engines for three of these languages.

Main contributions

- This paper presents an LLMC-based approach to multilingual repair that enables a flipped interaction model for AI-assisted programming in which the user writes code and the assistant suggests fixes.

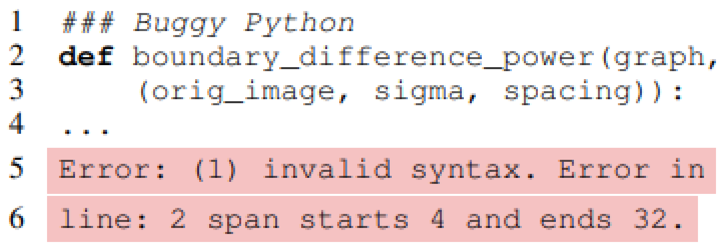
- The authors have implemented this approach in the RING system, which employs compiler (or diagnostic) messages, smart few shot selection, and ranking of repair candidates to perform repair across varying languages.

- The authors have performed an extensive evaluation across six different languages, showing that multilingual repair with LLMCs is viable and can compete with or outperform languag-especifc repair engines.

Procedure

The procedure has been divided into three portions. They are fault localisation, code transformation, and candidate ranking.

First step is fault localisation.Here, locating syntactic mistakes and some semantic errors, such as type errors, is aided by tools like the compiler, static analyzers, or linters. Example of such localisation is shown in the figure below.

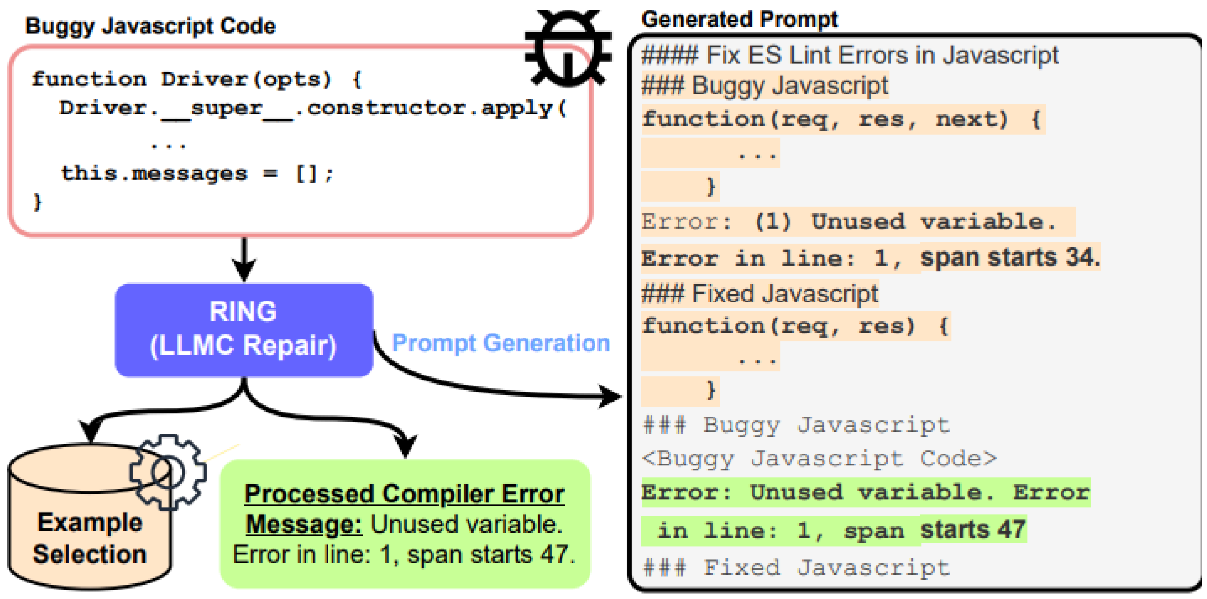


```
### Buggy Python
def boundary_difference_power(graph,
    (orig_image, sigma, spacing)):
...
Error: (1) invalid syntax. Error in
line: 2 span starts 4 and ends 32.
```

Fig 2: Bug localisation. Highlighting corresponds to the prepared syntax error message.

This error message is used as a few shot prompt in the paper.

With these error messages and error location, a prompt is generated, which is shown in the figure below.



```
function Driver(opts) {
  Driver.__super__.constructor.apply(
      ...
  this.messages = [];
}
```

```
#### Fix ES Lint Errors in Javascript
### Buggy Javascript
function(req, res, next) {
      ...
    }
Error: (1) Unused variable.
Error in line: 1, span starts 34.
### Fixed Javascript
function(req, res) {
      ...
    }
### Buggy Javascript
<Buggy Javascript Code>
Error: Unused variable. Error
 in line: 1, span starts 47
### Fixed Javascript
```

Fig 3: Prompt generation

Second step is code transformation through few-shot learning. It has been shown that LLMs are capable of few-shot learning—the ability to learn from a few examples of the intended task—by adding related examples of the task to the prompt. Now, based on the prompt, examples from a collection of buggy-fixed pairs based on error message similarity is selected. This collection of buggy-fixed pairs is called the example bank.
Message embedding selection method is used for selecting programs from the example bank.

Last step is candidate ranking. This paper has shown that multiple candidates for bug fix can be generated by controlling the temperature parameter during code generation. The final step in RING is to rank the candidates obtained by querying Codex using the prompt described in the prior two stages. By averaging the log probabilities of tokens selected during the decoding process and sorting the candidates in descending order of their averages, the suitable candidate is selected.

Evaluation Metrics

RQ1. Viability of Multilingual Repair:

RING vs. Language-Specific Engines:

- Excel: RING outperforms with a rate of 0.82 .

- Python: RING achieves a rate of 0.94.
- C: RING leads with a rate of 0.63.
- Power Fx: RING's rate (0.85) is comparable to LaMirage's .
- JavaScript: TFix on original snippets performs better than RING .
- PowerShell: RING underperforms with a rate of 0.18, attributed to the scarcity of PowerShell commands in Codex’s training data.

RQ2. Error Localization:

- **Localization Accuracy:**
    - RING locates errors more accurately than language-specific baselines.
    - Performance varies by language but can reach up to a quarter of unrepaired programs.
- **Program Length:**
    - Successful repairs correlate with shorter program lengths in JavaScript and Python.
    - No strong correlation observed for Excel due to its restrictive grammar.

RQ3. Code Transformation:

- **Performance Improvement:**
    - Excel: From 0.76 to 0.82 (8% increase).
    - JavaScript: From 0.43 to 0.46 (7% increase).
    - Python: From 0.91 to 0.94 (3% increase).
    - C: From 0.50 to 0.58 (16% increase).
    - PowerShell: From 0.15 to 0.18 (20% increase).

Key Learning And Finding from the Paper

This paper provides knowledge of prompt engineering. Prompt engineering in a tool which can leverage the power of LLMs without fine-tuning of user specific dataset. Prompt engineering has different techniques such as few shot prompts, zero shot prompts, etc. This paper also gives insights on error location using the compiler's message.

3) Title: TFix: Learning to Fix Coding Errors with a Text-to-Text Transformer

Authors: **Berkay Berabi** , **Jingxuan He**, **Veselin Raychev**, **Martin Vechev**

DOI: PMLR 139:780-791, 2021

This paper proposes an idea of using a pre-trained model on general NLP tasks and fine-tuning it on a specific dataset for fixing bugs in the code. It uses Text-to-Text Transfer Transformer (T5) as the pre-trained model and the dataset for fine-tuning is collected by extracting from the git commits. The author has given a name for their model as "TFix". TFix leverages the power of a pre-trained model on NLP by fine-tuning it on a specific task of finding coding bugs.

Pre-trained model: T5 (huggingface.co)
Code base link: eth-sri/TFix (github.com)

Procedure

This model is built for java scripts only but the authors have mentioned that it can be used for any programming language as this model (TFix) is built on a T5 transformer which is trained on general NLP tasks. Only fine-tuning affects the choice of programming language.

For the first part, the user sends a code written in java script. The code is sent to a static analyser for java, **ESLint**. ESLint checks the code snippet and generates an error message if any type of error is present in the snippet. This error message also contains the location of the error e.g., code lines.
The way TFix proposes the correct fix is by first extracting the error context consisting of the error line and the two neighbouring lines. Next, TFix represents all information about the error as a single piece of text as

fix **error type error message** error line : **error context**

Then TFix uses this text to query the pre-trained and fine-tuned model which generates new text representing the code that fixes the error. TFix can also fix multiple errors in a single attempt.

The main engine of this model is given below.

The authors have assumed the following parameters:

T → set of multiple errors
Lk → kth line in the program
M → error message
L → error context

Give all those parameters TFix generates the following template

text(e) = “fix”_t_ m_ lk_ “:”_ L

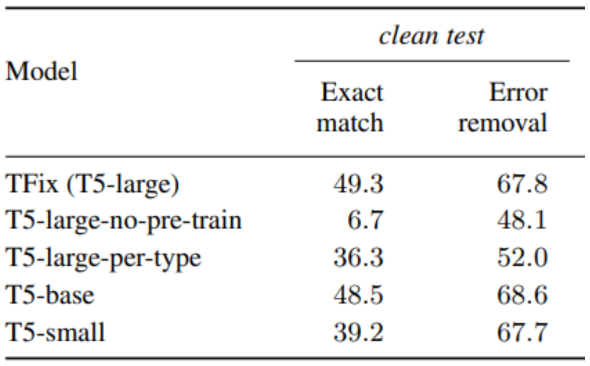
TFix queries a text-to-text encoder-decoder model which outputs L’ as text. L’ is used to replace the lines L in the code to fix the error e.

Evaluation Metrics

The authors have proposed two metrics for measuring the accuracy of TFix. They are **Exact Match** and **Error Removal.**

Exact match considers a prediction to be correct if and only if the fix perfectly matches the human fix done in the commit and Error removal counts a prediction as correct if the error is removed and no new error is introduced after the erroneous code is replaced by the prediction.

The performance of various variants of the model is shown below.



Model	*clean test*	
	Exact match	Error removal
TFix (T5-large)	49.3	67.8
T5-large-no-pre-train	6.7	48.1
T5-large-per-type	36.3	52.0
T5-base	48.5	68.6
T5-small	39.2	67.7

Key Learning And Finding from the Paper

This paper set a perfect example showing that a model trained on any raw data can be used to a particular task by fine-tuning it on a specific datasets. E.g., in this paper the authors have used a pre-trained model which trained on basic NLP tasks rather than on code. But, fine-tuning it on a specific dataset consisting of error and fix can make the model generate the correct code.

4) Title: Deep Neural Solver for Math Word Problems.

Authors: **Yan Wang, Xiaojiang Liu, ShumingShi**
Tencent AI Lab

DOI: https://doi.org/10.18653/v1/D17-1088

This paper focuses on the development of a deep neural solver for automatic maths problem solving. Using RNN, maths word problems are converted to equation templates. The main objective of this paper is to develop a hybrid model that combines the RNN model and a similarity-based retrieval model to get better performance and higher accuracy. This model outperforms the state-of-art model statistical method of maths words problem solving.

Main contributions

- The first contribution in this paper is using Deep Neural Network (DNN) technology for automatic maths word problem solving. According to the authors, this is the first work to apply DNN techniques to this particular problem domain.

- The second contribution is the proposal of a hybrid model. This model combines a sequence-to-sequence (seq2seq) approach with a similarity-based retrieval model. The authors claim that this combination leads to further performance improvements in solving maths word problems.

- The third significant contribution is the construction of a large dataset. The authors developed this dataset specifically to facilitate and advance the study of automatic maths problem solving. This resource likely provides a valuable benchmark for future research in this area.

Procedure

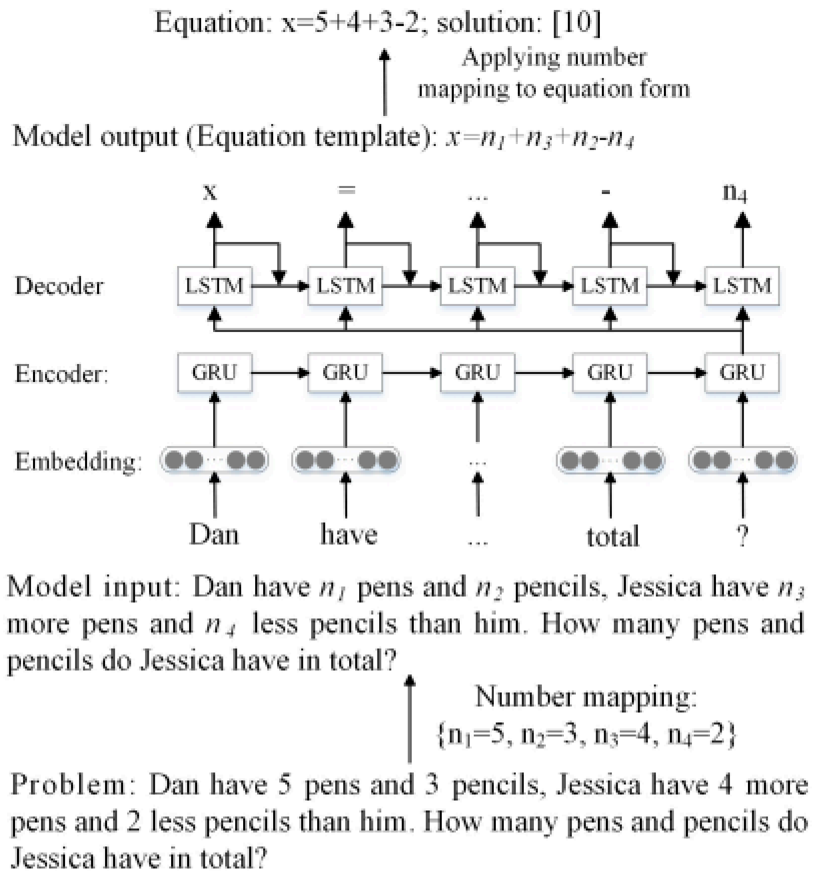
1) Problem Formulation:
- The first step in this paper is to define a maths word problem P as a word sequence Wp with a set of variables Vp.
- Then, it maps the equation Ep to an equation template Tp using a number mapping Mp.

2) Dataset Construction:
- Authors have crawled over 60,000 Chinese maths word problems from online education websites.
- They focus on one-unknown-variable linear maths word problems.
  Result: Math23k dataset with 23,161 labelled problems.

dataset	# problems	# templates	# sentences	# words	problem types
Alg514	514	28	1.62k	19.3k	algebra, linear
Dolphin1878	1,878	1,183	3.30k	41.4k	number word problems
DRAW-1K	1,000	Unknown	6.23k	81.5k	algebra, linear, one-VAR
Math23K	23,161	2,187	70.1k	822k	algebra, linear, one-VAR

Figure: Statistics of MAth23k dataset and other publicly available datasets

3) RNN-based Seq2seq Model:
- Input: Problem text after number mapping.
- Output: Equation template.
- Architecture:
  a. Word embedding layer
  b. Two-layer GRU as encoder
  c. Two-layer LSTM as decoder
- A custom activation function with predefined rules is used to ensure mathematically correct outputs.



Equation: x=5+4+3-2; solution: [10]

Model output (Equation template): $x=n_1+n_3+n_2-n_4$

Model input: Dan have n_1 pens and n_2 pencils, Jessica have n_3 more pens and n_4 less pencils than him. How many pens and pencils do Jessica have in total?

Number mapping: $\{n_1=5, n_2=3, n_3=4, n_4=2\}$

Problem: Dan have 5 pens and 3 pencils, Jessica have 4 more pens and 2 less pencils than him. How many pens and pencils do Jessica have in total?

Fig: Seq2seq model

4) Significant Number Identification (SNI):

- Then, a LSTM-based binary classification model is developed. The purpose of this model is to determine whether a number in the problem text should appear in the equation.
- Training data for SNI model is extracted from maths word problems and each number is labelled as significant or insignificant.

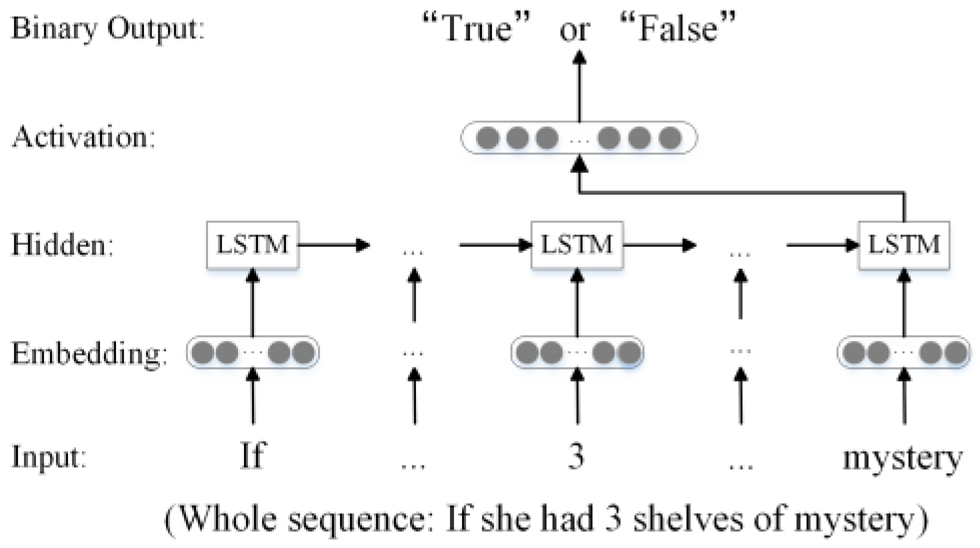


Fig: Significant number identification model

5) Similarity-based Retrieval Model:
- The retrieval model solves maths word problems by calculating the lexical similarity between the testing problem and each problem in the training data using TF-IDF scores and Jaccard similarity.
- The model then applies the equation template of the most similar problem to the testing problem, with an observed correlation between higher similarity scores and increased solution accuracy, which is leveraged in the hybrid model.

6) Hybrid Model:
- In this model, the seq2seq model and the retrieval model are combined.
- A threshold on the similarity score is used to decide which model to use for each problem.

7) Training and Evaluation:
- Training the models on the Math23k dataset, evaluating the performance using accuracy metrics and comparing results of the RNN model, retrieval model, and hybrid model against state-of-the-art statistical learning methods are the final procedure of this paper.

Key findings from the paper:

I learned how deep neural networks, particularly RNN-based sequence-to-sequence models, can be applied to solve maths word problems by translating problem text into equation templates. The paper provided a clear explanation of the problem formulation, including the process of number mapping and equation template generation. It also gave insights into the importance of large datasets for training deep learning models in this domain, as evidenced by the creation of the Math23k dataset.

The paper offered a comprehensive explanation of the entire process, from problem text input to equation generation, including the novel approach of Significant Number Identification (SNI). Additionally, it provided valuable insights into combining different modelling techniques through the proposed hybrid model.

The most important finding is that the hybrid model, combining the seq2seq approach with a similarity-based retrieval model, outperforms either model alone. This highlights the complementary strengths of different approaches in solving maths word problems. The seq2seq model showed good generalization ability on unseen equation templates, while the retrieval model performed well on problems highly similar to those in the training set.

Another crucial finding is the importance of the similarity score threshold in the hybrid model. This threshold determines which model (seq2seq or retrieval) is used for each problem, significantly impacting overall performance. The positive correlation between similarity scores and retrieval model accuracy provides a solid basis for this hybrid approach.

The paper also demonstrates that while deep learning models can significantly outperform traditional statistical learning methods on large, diverse datasets, there's still room for improvement. This suggests potential for future research in enhancing model architectures and incorporating more sophisticated features to further improve performance in automatic maths word problem solving.

5) Title: End-to-End Open-Domain Question Answering with BERTserini

Authors: Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li,
Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin
David R. Cheriton School of Computer Science, University of Waterloo
2 RSVP.ai
DOI: http://dx.doi.org/10.18653/v1/n19-4013

The paper introduces a novel system that integrates BERT with the Anserini information retrieval toolkit to address the limitations of current question answering (QA) and reading comprehension models. Traditional QA models operate on small text inputs, whereas BERTserini is designed to work end-to-end on a large corpus of Wikipedia articles, thereby leveraging best practices from information retrieval (IR) to improve accuracy and performance in identifying answer spans.

Main contributions:

- The paper presents BERTserini, an end-to-end open-domain question answering system that integrates BERT with the open-source Anserini information retrieval toolkit. Unlike most current QA models that operate on small amounts of input text, BERTserini combines IR best practices with a BERT-based reader to find answers from a large corpus of Wikipedia articles in an end-to-end manner.
- The system demonstrates significant improvements over previous results on a standard benchmark test collection.
- The authors show that fine-tuning pretrained BERT with SQuAD is sufficient to achieve high accuracy in identifying answer spans.

System architecture:

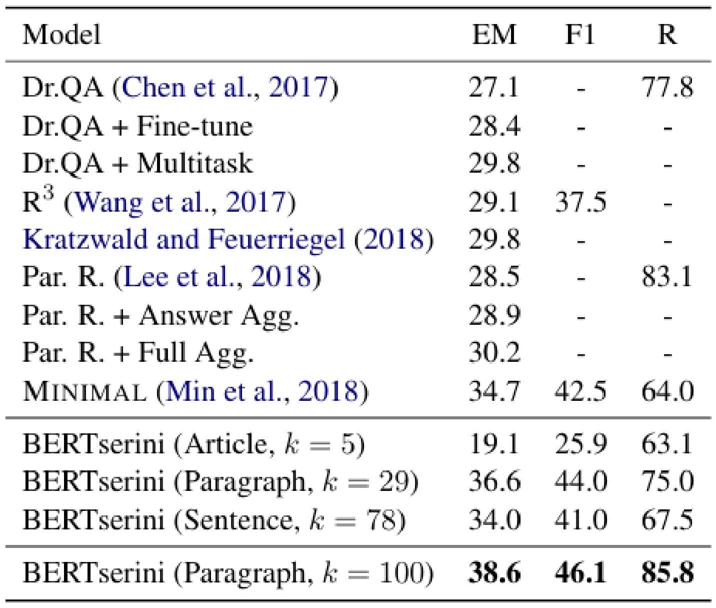
BERTserini consists of two main modules:

Anserini Retriever: Responsible for selecting text segments that may contain the answer. The authors experimented with different granularities of text indexing.They are Article-level, Paragraph-level and Sentence-level.

BERT Reader: Identifies answer spans within the retrieved text segments. It uses a fine-tuned BERT model based on the SQuAD dataset.

Key findings:

- Paragraph-level retrieval outperforms both article-level and sentence-level retrieval. The system achieves state-of-the-art results on the SQuAD dataset for open-domain QA, with significant improvements over previous work.
- Increasing the number of retrieved paragraphs (k) improves performance up to a point, but with diminishing returns after k=10.



Model	EM	F1	R
Dr.QA (Chen et al., 2017)	27.1	-	77.8
Dr.QA + Fine-tune	28.4	-	-
Dr.QA + Multitask	29.8	-	-
R^3 (Wang et al., 2017)	29.1	37.5	-
Kratzwald and Feuerriegel (2018)	29.8	-	-
Par. R. (Lee et al., 2018)	28.5	-	83.1
Par. R. + Answer Agg.	28.9	-	-
Par. R. + Full Agg.	30.2	-	-
MINIMAL (Min et al., 2018)	34.7	42.5	64.0
BERTserini (Article, $k = 5$)	19.1	25.9	63.1
BERTserini (Paragraph, $k = 29$)	36.6	44.0	75.0
BERTserini (Sentence, $k = 78$)	34.0	41.0	67.5
BERTserini (Paragraph, $k = 100$)	**38.6**	**46.1**	**85.8**

Fig: EM, F1 and R score with different Anserini retrieval conditions

- Error analysis reveals that passage retrieval is not the primary bottleneck in the current implementation. The main areas for improvement are in answer extraction and score aggregation.
- The system demonstrates practical latency, with average processing times of 0.5s for retrieval and 0.18s for BERT processing per question.

Significance:
This paper represents an important step in combining traditional IR techniques with modern neural language models for open-domain question answering. By integrating BERT with Anserini, the authors have created a system that can effectively search large document collections and extract precise answers, outperforming previous approaches. The simplicity and effectiveness of the architecture make it a promising foundation for future research in this area.

The authors have also deployed BERTserini as an interactive chatbot, demonstrating its practical applicability beyond just benchmark evaluations. This work highlights the potential of end-to-end neural approaches in creating more effective and user-friendly QA systems.

6) **Title: SAAS: Solving Ability Amplification Strategy for Enhanced Mathematical Reasoning in Large Language Models**

Authors: **Hyeonwoo Kim, Gyoungjin Gim, Yungi Kim, Jihoo Kim, Byungju Kim, Wonseok Lee, Chanjun Park**

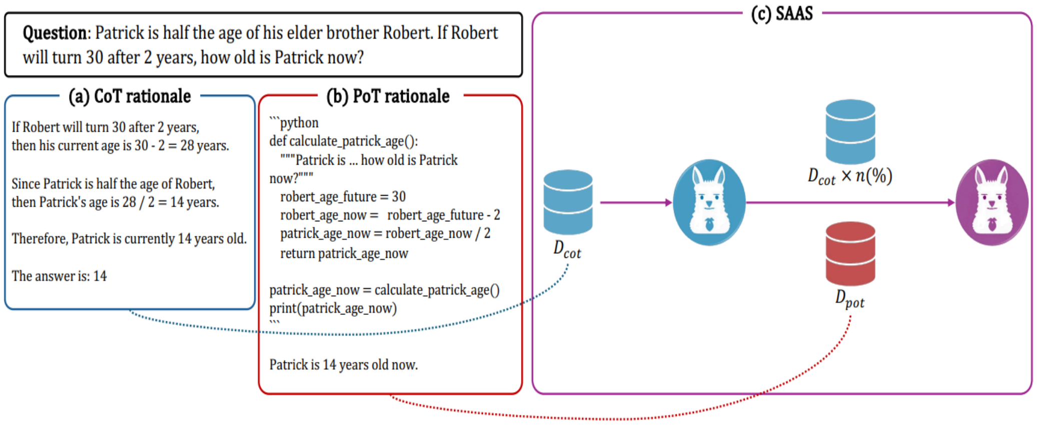
DOI: https://doi.org/10.48550/arXiv.2404.03887

The focus of this paper is to integrate Chain-of-Thought (CoT) and Program-of-Thought (PoT) learning. It is hypothesised that the learning of mathematical reasoning ability is helpful for the amplification of problem-solving-ability. The authors propose a sequential learning approach namely SAAS (Solving Ability Amplification Strategy) which strategically transits from CoT to PoT learning.

Research Questions:

- RQ1: Does SAAS quantitatively outperform its competitors for solving challenging mathematical problems?
- RQ2: Are two core strategies of SAAS (sequential learning, cognitive retention strategy) effective in improving the accuracy?
- RQ3: Is SAAS effective in solving not only basic but also challenging mathematical problems?
- RQ4: Does sequential learning that transitions from CoT learning to PoT learning help improve both the mathematical reasoning and computational accuracy?

Procedure:



Question: Patrick is half the age of his elder brother Robert. If Robert will turn 30 after 2 years, how old is Patrick now?

(a) CoT rationale

If Robert will turn 30 after 2 years, then his current age is 30 - 2 = 28 years.

Since Patrick is half the age of Robert, then Patrick's age is 28 / 2 = 14 years.

Therefore, Patrick is currently 14 years old.

The answer is: 14

(b) PoT rationale

~~~
```python
def calculate_patrick_age():
    """Patrick is ... how old is Patrick
now?"""
    robert_age_future = 30
    robert_age_now =  robert_age_future - 2
    patrick_age_now = robert_age_now / 2
    return patrick_age_now

patrick_age_now = calculate_patrick_age()
print(patrick_age_now)
```
~~~

Patrick is 14 years old now.

Fig: Overview of SAAS with two core Strategy i) sequential learning strategy ii) cognitive retention strategy

The paper focuses on a fine-tuning approach. CodeLLaMA 13B model was used as a base model and fine-tuned. Accuracy was calculated as metrics for the final answer after mathematical computation.

Analysis:

1. Simply combining CoT and PoT learning does not effectively solve complex mathematical problems.
2. only CoT learning approach leads to arithmetic calculation errors;
    a. Only PoT learning approach may result in a deficit of mathematical reasoning;
    b. Sequential learning that transitions from CoT learning to PoT learning helps improve computational accuracy as well as mathematical reasoning.

Datasets:

1. GSM8K (Cobbe et al., 2021)
2. MATH (Hendrycks et al., 2021)
3. MetaMathQA (Yu et al., 2023)
4. MathInstruct (Yue et al., 2023)
5. QANDA

Key Findings from the Paper:

1. Prioritising the learning of mathematical reasoning ability via Chain-of-Thought (CoT) learning is helpful for the amplification of problem-solving ability during Program-of-Thought (PoT) learning.
2. For effective sequential learning, it is necessary to employ a cognitive retention strategy.

7) Title: ChatGLM-Math: Improving Math Problem-Solving in Large Language Models with a Self-Critique Pipeline

Authors: **Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Wenyi Zhao, Jie Tang, Yuxiao Dong**

DOI: https://doi.org/10.48550/arXiv.2404.02893

The main goal of the paper is to tailor the self-critique pipeline. This method scores mathematical responses generated by models based on questions and reference answers, including an output of explanatory analysis and a score between 1 and 10. It is essential to note that the model's primary language is Chinese.

Contributions:

1. Newly created challenging dataset, MATHUSEREVAL. Related evaluation dataset and scripts are released at https://github.com/THUDM/ChatGLM-Math.
2. Related techniques have been deployed to https://chatglm.cn.

3. Introduction of the Self-Critique pipeline, a novel framework that elevates both the mathematical and linguistic capabilities of LLMs through self-generated feedback, thereby eliminating the need for external supervisory models and manual annotations.

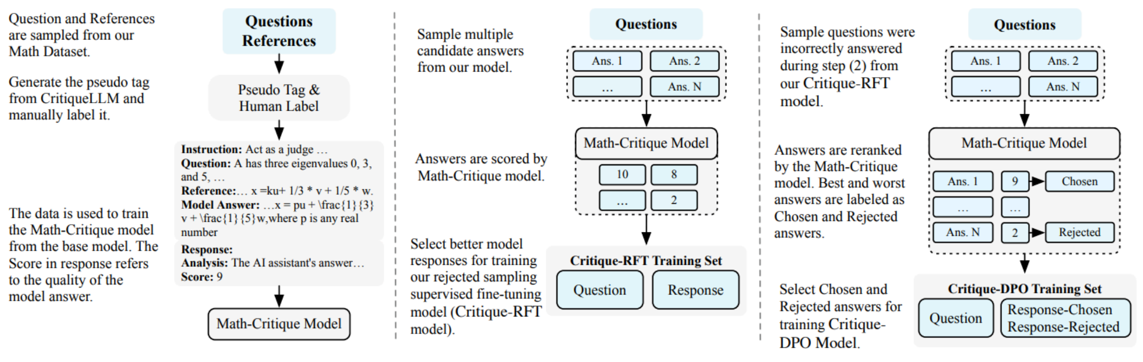
Procedure:

Self-Critique pipeline is a weakly supervised iterative training method for enhancing mathematical abilities, originating from a single model. Initially, a Math-Critique model is trained using the base model.

1. Math-Critique training:
    a. Compared to traditional reward models, this approach leverages the contextual capabilities of language models, enabling more accurate judgments by integrating reference answers.
    b. MathCritique(Question, Reference, Answer) → (Critique, Score)

2. Critique-RFT training:
    This utilises a rejection sampling method based on Math-Critique. It consists of re-examined and redesigned the implementation of RFT and found that both the sampling range and the model influence the outcomes during the rejection sampling process

3. Critique-DPO Training:
    During the Critique-DPO phase, it was observed that the direct use of DPO loss led to instability in the training process. A cross-entropy loss for the chosen answer was introduced as a regularization term to the total loss to mitigate this issue.



Question and References are sampled from our Math Dataset.

Generate the pseudo tag from CritiqueLLM and manually label it.

Instruction: Act as a judge …
Question: A has three eigenvalues 0, 3, and 5, …
Reference:… x =ku+ 1/3 * v + 1/5 * w.
Model Answer: …x = pu + \frac{1}{3} v + \frac{1}{5}w,where p is any real number

Response:
Analysis: The AI assistant's answer…
Score: 9

The data is used to train the Math-Critique model from the base model. The Score in response refers to the quality of the model answer.

Sample multiple candidate answers from our model.

Answers are scored by Math-Critique model.

Select better model responses for training our rejected sampling supervised fine-tuning model (Critique-RFT model).

Sample questions were incorrectly answered during step (2) from our Critique-RFT model.

Answers are reranked by the Math-Critique model. Best and worst answers are labeled as Chosen and Rejected answers.

Select Chosen and Rejected answers for training Critique-DPO Model.

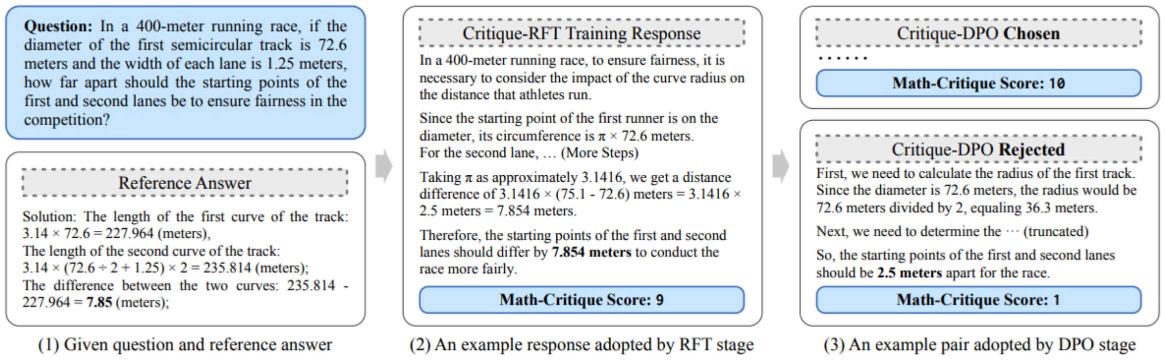
(1) Train Math-Critique (2) Train Critique-RFT Model (3) Train Critique-DPO Model

Fig: Self-Critique pipeline for ChatGLM-Math

Since most of the work is conducted in Chinese, the authors selected three categories of baselines: open source mathematics-specific models, open-source Chinese models, and leading proprietary models.

Datasets:

- MATHUSEREVAL dataset
- English academic datasets: GSM8k, MATH
- Chinese academic datasets: ape210k, cmath
- Out-Of-Distribution test set: Hungarian National Exam



Question: In a 400-meter running race, if the diameter of the first semicircular track is 72.6 meters and the width of each lane is 1.25 meters, how far apart should the starting points of the first and second lanes be to ensure fairness in the competition?

Reference Answer

Solution: The length of the first curve of the track: $3.14 \times 72.6 = 227.964$ (meters),
The length of the second curve of the track:
$3.14 \times (72.6 \div 2 + 1.25) \times 2 = 235.814$ (meters);
The difference between the two curves: $235.814 - 227.964 =$ **7.85** (meters);

(1) Given question and reference answer

Critique-RFT Training Response

In a 400-meter running race, to ensure fairness, it is necessary to consider the impact of the curve radius on the distance that athletes run.

Since the starting point of the first runner is on the diameter, its circumference is $\pi \times 72.6$ meters.
For the second lane, … (More Steps)

Taking π as approximately 3.1416, we get a distance difference of $3.1416 \times (75.1 - 72.6)$ meters $= 3.1416 \times 2.5$ meters $= 7.854$ meters.

Therefore, the starting points of the first and second lanes should differ by **7.854 meters** to conduct the race more fairly.

Math-Critique Score: 9

(2) An example response adopted by RFT stage

Critique-DPO **Chosen**

……

Math-Critique Score: 10

Critique-DPO **Rejected**

First, we need to calculate the radius of the first track. Since the diameter is 72.6 meters, the radius would be 72.6 meters divided by 2, equaling 36.3 meters.

Next, we need to determine the ⋯ (truncated)

So, the starting points of the first and second lanes should be **2.5 meters** apart for the race.

Math-Critique Score: 1

(3) An example pair adopted by DPO stage

Fig: Training Dataset Example

8) **Title: Large Language Models for Mathematical Reasoning: Progresses and Challenges**

Authors: **Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, Wenpeng Yin**

DOI: https://doi.org/10.48550/arXiv.2402.00157

This paper stands as one of the first extensive examinations of the landscape of LLMs in the realm of mathematics, providing a holistic perspective on the current state, accomplishments, and future challenges in this rapidly evolving field. This survey covers four crucial dimensions: a meticulous exploration of math problem types and the datasets associated with them; an in-depth analysis of the evolving techniques employed by LLMs in mathematical problem-solving; an examination of factors that affect the LLMs solving math problems; and a critical discussion on the persisting challenges that loom over this burgeoning field.

Datasets:

1. Arithmetic: MATH-140
2. Math Word Problems (MWP):
    a. Question-Answer: CMATH, SAT-MATH
    b. Question-Equation-Answer: SVAMP, MULTIARITH, MAWPS, etc.

    c. Question-Rationale-Answer: MATH, PRM800K, MATHQA, AQUA, LILA, MATH-INSTRUCT, etc.
3. Geometry: UNIGEO, GEOQA, GEOMETRY 3K, etc.
4. Automated theorem proving: MINIF2F, COQGYM, etc.
5. Math in vision-language context: CHARTQA, MATHVISTA.

Methodologies:

- Categorized into three levels of methods:
  i) Prompting frozen LLMs, ii) Strategies enhancing frozen LLMs, and iii) Fine-tuning LLMs.

(i) Prompting frozen LLMs: prompting methods like vanilla prompt, Program-of-Thoughts prompt, Program Synthesis prompt, Chain-of-Thought etc.

(ii) Strategies enhancing frozen LLMs: Chain-of-Thought, Self-Consistency, explicit code based self-verification, using external tools,

(iii) Fine-tuning LLMs

Factors in Evaluation:
While accuracy is crucial in evaluating LLMs for maths problem-solving, it shouldn't be the sole metric. Other important dimensions include:

1. Confidence Provision: Imani et al. (2023)'s "MathPromper" boosts LLM performance and confidence by generating algebraic expressions, providing diverse prompts, and evaluating consensus among multiple runs.
2. Verifiable Explanations: Gaur and Saunshi (2023) used concise, verifiable explanations to assess LLM reasoning, revealing their proficiency in zero-shot solving of symbolic MWP and their ability to produce succinct explanations.

9) **Title: Llemma: An Open Language Model For Mathematics**

Authors: **Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, Sean Welleck**

DOI: https://doi.org/10.48550/arXiv.2310.10631

The paper presents LLEMMA (7B & 34B parameters model), a doman-specific large language model for mathematics. This model is yielded as a result of continued pretraining of

Code Llama on Proof-Pile-2 dataset, which is capable of tool use and formal theorem proving without any further fine tuning.

Major Contributions:

a. Release of LLEMMA models: 7B and 34B parameters models for mathematics.
b. AlgebraicStack, an 11B-token dataset of source code from 17 languages, spanning numerical, symbolic, and formal math.
c. Open source training data and code: https://github.com/EleutherAI/math-lm

Datasets:

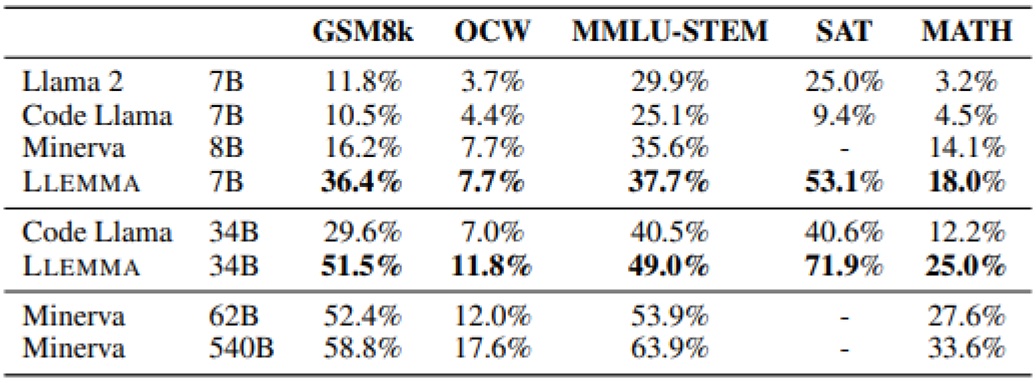
- AlgebraicStack
- OpenWebMath: a 15B-token dataset of high-quality web pages filtered for mathematical content.
- ArXiv subset of RedPajama: an open-access reproduction of the LLaMA training dataset. The ArXiv subset contains 29B tokens.
- Proof-Pile-2

Evaluation:

- Compare llemma models using few shot evaluations.
- First, the model's ability to solve mathematics problems was evaluated using a chain of thought reasoning (Wei et al., 2023) and majority voting (Wang et al., 2023). Then, few shot tools use evaluations.
- CHAIN-OF-THOUGHT MATHEMATICAL PROBLEM SOLVING:

  These tasks involve generating self-contained text solutions to problems expressed in LATEX or natural language, without using external tools.

  a. MATH: Given a problem statement, the model generates a LATEXsolution and an answer that must match a reference answer.
  b. GSM8k: Used the 8-shot prompt from Wei et al. (2023).

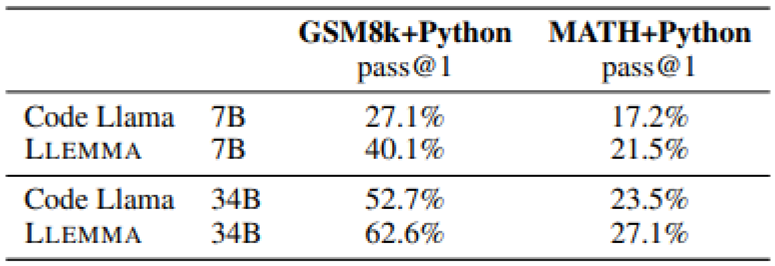


		GSM8k	**OCW**	**MMLU-STEM**	**SAT**	**MATH**
Llama 2	7B	11.8%	3.7%	29.9%	25.0%	3.2%
Code Llama	7B	10.5%	4.4%	25.1%	9.4%	4.5%
Minerva	8B	16.2%	7.7%	35.6%	-	14.1%
LLEMMA	7B	**36.4%**	**7.7%**	**37.7%**	**53.1%**	**18.0%**
Code Llama	34B	29.6%	7.0%	40.5%	40.6%	12.2%
LLEMMA	34B	**51.5%**	**11.8%**	**49.0%**	**71.9%**	**25.0%**
Minerva	62B	52.4%	12.0%	53.9%	-	27.6%
Minerva	540B	58.8%	17.6%	63.9%	-	33.6%

Table: Results on chain-of-thought reasoning tasks

- MATHEMATICAL PROBLEM SOLVING WITH TOOL USE: The final answer is a program that executes to a numeric type or a SymPy object. Few-shot prompts include examples that use built-in numeric operations, the math module, and SymPy.

a. MATH + Python
b. GSM8k + Python



		GSM8k+Python pass@1	**MATH+Python** pass@1
Code Llama	7B	27.1%	17.2%
LLEMMA	7B	40.1%	21.5%
Code Llama	34B	52.7%	23.5%
LLEMMA	34B	62.6%	27.1%

Table: Mathematical Problem-solving with tool use

- LLEMMA improves over Code Llama on both tasks. Its performance on MATH and GSM8k with tools is also higher than its performance on these datasets without tools.

Literature Review Matrix

S.N.	Title/Author/ Date	Conceptual Framework	Research Question(s)/ Hypotheses	Datasets	Methodology	Analysis & Results	Conclusions	Implications for Future Research
1.	Can we learn from developer mistakes? Learning to localise and repair real bugs from real bug fixes. **Cedric Richter & Wehrheim (July 2022)**	Code debugger model (RealiT), Finding bug location and generating solution of the bugs in a step wise manner with explanation.	Can a model pre-trained on code mutants and fine-tuned on real bug fixes localise and repair more code bugs than traditional models like RNN, GNN, and GREAT?	Not provided	code debugger model (RealiT), which is pre-trained on a large number of artificial bugs produced by traditional mutation operation and fine tuned on a smaller set of real bug fixes.	RealiT showed significant improvements in localization and repair accuracy compared to the baseline models, particularly when fine-tuned on real bug fixes.	Only pre-training on code mutants does not give a good accuracy model because the model is trained only on code mutants, which may differ from the actual mistake made by human programmers. So, fine-tuning on real code	Implementation of RealiT on different language models and different error types

							bug fixes is a must.	
2.	Repair Is Nearly Generation: Multilingual Program Repair with LLMs **Harshit Joshi et al. (June 2023)**	Code repair engine (RING) is developed. prompt engineering is used in order to localisation and transformation of the bugs.	Can prompt engineering perform better when used in code repair?	No dataset	Fault localisation, prompt generation, solution generation.	The authors have performed an extensive evaluation across six different languages, showing that multilingual repair with LLMCs is viable and can compete with or outperform languag-especifc repair engines.	Prompt engineering can perform as better as fine tuning	Developi ng an user friendly chat bot for code repair.
3.	TFix: Learning to Fix Coding Errors with a Text-to-Text Transformer. **Berkay Berabi et al.**	Text to text transformer pre-trained LLM model is fine tuned on specific dataset.	Can a model trained on any raw data be used to a particular task by fine-tuning it on a specific datasets?	eth-sri/TFi x (github.co m)	Text to text transformer pre-trained transformer model (T5) fine tuned on specific dataset consisting of bugs and its fixes of java	Tfix model performed well with 67.8% of error removal.	Power of pre-trained models can be leveraged by fine tuning the model for specific tasks.	Implemen tation of model on different program ming languages .

					code.			
4.	Deep Neural Solver for Math Word Problems **Yan Wang et al. (September 2017)**	Develop a hybrid model that combines the RNN model and a similarity-based retrieval model to get better performance and higher accuracy.	Whether a RNN model can effectively solve math word problems by translating them into equation templates.	Math23k dataset	RNN model and a hybrid model combining RNN with a similarity-based retrieval model	The RNN model and hybrid model outperformed state-of-the-art statistical learning methods on the Math23k dataset.	Deep neural networks, particularly the hybrid model, show promise in solving math word problems more accurately than traditional methods.	Further exploration of different neural network architectures and improvements in dataset size and diversity.
5.	End-to-End Open-Domain Question Answering with BERTserini **Wei Yang et al. (June 2019)**	Integration of BERT with Anserini information retrieval toolkit to improve open-domain QA performance.	How can integrating BERT with information retrieval improve QA performance?	SQuAD dataset	Integration of BERT (deep learning model) with Anserini (information retrieval toolkit); fine-tuning BERT with SQuAD for enhanced performance.	Demonstrated significant improvements in QA performance with fine-tuning BERT on SQuAD; high accuracy in answer retrieval from Wikipedia.	Combining BERT with robust IR tools enhances performance in open-domain QA; fine-tuning with SQuAD is effective for high accuracy.	Future research can optimize BERT-IR integration and apply this approach to other large-scale datasets and QA tasks.

6.	SAAS: Solving Ability Amplification Strategy for Enhanced Mathematical Reasoning in Large Language Models **Zhangir Azerbayev et al.** (April 2024)	Sequential learning approach namely SAAS (Solving Ability Amplification Strategy)	It is hypothesised that the learning of mathematical reasoning ability is helpful for the amplification of problem-solving-ability.	GSM8K, MATH, MetaMathQA, MathInstruct, QANDA	Fine-tuning CodeLLaMA 13B model integrating Chain-of-Thought (CoT) and Program-of-Thought (PoT) learning.	Sequential learning that transitions from CoT learning to PoT learning helps improve computational accuracy as well as mathematical reasoning.	Prioritising the learning of mathematical reasoning ability via CoT learning is helpful for the amplification of problem-solving ability during PoT learning.	Exploration of SAAS in other fields.
7.	ChatGLM-Math: Improving Math Problem-Solving in Large Language Models with a Self-Critique Pipeline **Janice Ahn et al.** (April 2024)	Self-Critique pipeline, a novel framework that elevates both the mathematical and linguistic capabilities of LLMs.	How to simultaneously maintain and improve both language and mathematical capabilities in deployed LLM?	GSM8k, MATH, MATHUSEREVAL, ape210k, cmath	Math-Critique training, Critique-RFT training, Critique-DPO Training.	Suggested pipeline significantly enhances the LLM's mathematical problem-solving while still improving its language ability, outperforming LLMs that could be two times larger.	ChatGLM-Math, a 32-billion parameter model achieved state-of-the-art results among open-source language models on multiple datasets.	Implementation of graphic thinking and drawing abilities, and Precision calculation capability.

8.	Large Language Models for Mathematical Reasoning: Progresses and Challenges **Yifan Xu et al.** (April 2024)	Survey, providing a holistic perspective on the current state, accomplishments, and future challenges in the realm of mathematics.	What is the current state of LLM in mathematics, and what recent developments have occurred?	CMATH, SAT-MATH, MATH-140, SVAMP, MULTIARITH, MAWPS, MATH, PRM800K, MATHQA, AQUA, LILA, MATH-INSTRUCT, UNIGEO, GEOQA, GEOMETRY 3K, etc	Study on i) Prompting frozen LLMs, ii) Strategies enhancing frozen LLMs, and iii) Fine-tuning LLMs.	Accuracy is crucial in evaluating LLMs for maths problem-solving, it shouldn't be the sole metric. Other important dimensions include Confidence Provision, and Verifiable Explanations.	The paper presented an overview on various aspects of LLMs in mathematical reasoning, including their capabilities and limitations.	Utilizing information provided for further research of LLM in the mathematical field.
9.	Llemma: An Open Language Model For Mathematics **Zhangir Azerbayev et al.**	Pretraining	How to enhance the capabilities of large language models in mathematical reasoning and	Algebraic Stack, Proof-Pile-2, OpenWeb Math, ArXiv	Continued pretraining of Code Llama on Proof-Pile-2 dataset.	Llemma demonstrates significant improvements over previous models in various benchmarks. For instance, llemma 34B achieved a	Llemma outperforms all known open base models, as well as the unreleased Minerva model suite on an equi-parameter	Long context finetuning procedure on trained LLEMMA 7B model

	(March 2024)		problem-solving by continuing pretraining on a specialized dataset ?	subset of RedPajama		20% point improvement over Code Llama on GSM8k.	basis.	

Project Solution Proposals:

For our project, we will be using prompt engineering and fine tuning techniques. For the maths Q&A, (and AI Q&A, if feasible) we will be fine tuning the pre-trained LLM model with the datasets which are collected by us. The pre-trained models will be small language models by microsoft, phi-3 and phi-2 (**These models may be changed if we get access to A100 or better GPUs)**. For the code debugger, we will be using the Mistral-7B-Instruct-v0.3 (might be changed as work progresses) model with prompt engineering. Also, we do not have enough resources to execute the user given code in a controlled environment to capture runtime errors. So, we will probably stick to just debugging python codes for now as we can execute python code and capture errors using exec() in google colab, or study about other alternatives later on with the development of the project. We are also exploring the possibility of using a mixture of agents for Code debugging and AI Q&A tasks.

To make the model more accurate and updated with the recent development of the world, Retrieval-Augmented Generation (RAG) can be used. The main objective of RAG is to enhance the model's responses by providing it with additional context from an external knowledge base. RAG is used in order to add more context to the prompt which can be passed to the model to get response from it. The basics block diagram of our model including fine tuning ang RAG is shown below.

Note: The following represents the proposed architecture of our system, with the understanding that minor adjustments may be made. It is also worth noting that RAG can be made optional considering various situations.

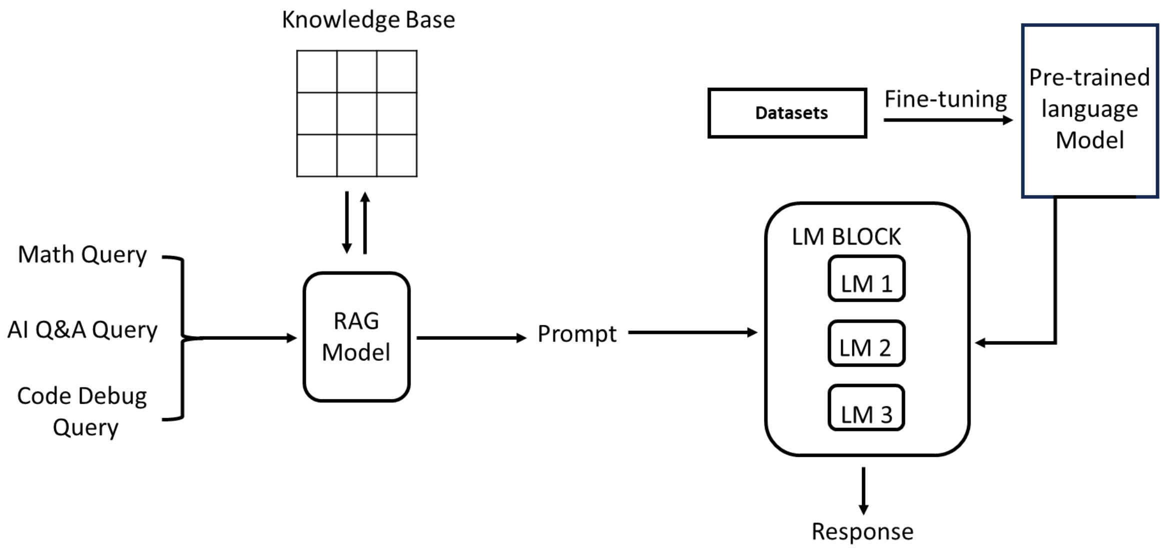


Fig: Project block diagram[1]

[1] LM BLOCK: This block in the diagram can consist of multiple LLMs, specific to different components of our project.

This is the general block diagram of the project. One thing to be noted is that we probably will not be fine tuning the code debugger model because of stint datasets.

For model evaluation, we are doing study as there's no strict way to evaluate any language models. Math Q&A may probably be evaluated based on accuracy, exact match of just the final numeric answer of our model with that of ground truth value. But this has its own limitations as it can not be applied to some mathematical problems involving theorem proving, etc. AI Q&A might be evaluated using other advanced LLM as a Judge, and code debugging on success rate of language model to debug on buggy test codes. Note that, we have not finalized our evaluation metrics yet and this is just a speculation.

References:

1. Richter, C., & Wehrheim, H. (2022). Can we learn from developer mistakes? Learning to localize and repair real bugs from real bug fixes. arXiv. https://arxiv.org/abs/2207.00301

2. Joshi, H., Cambronero Sanchez, J., Gulwani, S., Le, V., Verbruggen, G., & Radiček, I. (2023). Repair is nearly generation: Multilingual program repair with LLMs. Proceedings of the AAAI Conference on Artificial Intelligence, 37(4), 5131-5140. https://doi.org/10.1609/aaai.v37i4.25642

3. Berabi, B., He, J., Raychev, V., & Vechev, M. (2021). TFix: Learning to fix coding errors with a text-to-text transformer. In M. Meila & T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning (Vol. 139, pp. 780-791). PMLR. https://proceedings.mlr.press/v139/berabi21a.html

4. Wang, Y., Liu, X., & Shi, S. (2017). Deep neural solver for math word problems. In M. Palmer, R. Hwa, & S. Riedel (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 845-854). Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1088

5. Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., & Lin, J. (2019). End-to-end open-domain question answering with BERTserini. In W. Ammar, A. Louis, & N. Mostafazadeh (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations) (pp. 72-77). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4013

6. Kim, H., Gim, G., Kim, Y., Kim, J., Kim, B., Lee, W., & Park, C. (2024). SAAS: Solving ability amplification strategy for enhanced mathematical reasoning in large language models. arXiv. https://arxiv.org/abs/2404.03887

7. Xu, Y., Liu, X., Liu, X., Hou, Z., Li, Y., Zhang, X., Wang, Z., Zeng, A., Du, Z., Zhao, W., Tang, J., & Dong, Y. (2024). ChatGLM-Math: Improving math problem-solving in large language models with a self-critique pipeline. arXiv. https://arxiv.org/abs/2404.02893

8. Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R., & Yin, W. (2024). Large language models for mathematical reasoning: Progresses and challenges. arXiv. https://arxiv.org/abs/2402.00157

9. Azerbayev, Z., Schoelkopf, H., Paster, K., Dos Santos, M., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., & Welleck, S. (2024). Llemma: An open language model for mathematics. arXiv. https://arxiv.org/abs/2310.10631