Next-Gen Programming:

Maths Prompt Translator

Berlin

Boris Petreski, Ludvig Sanell, and Casper Kejser

Project objectives and scope

- Tasked with training model for code generation
- Our decision: mathematical queries into Python code

Development Journey

Choosing the model & initial testing

- Potential candidates
  - CodeGen
  - CodeT5
  - GPT-2
- What we were after
- Initial model testing
  - Dataset & code
- Results
- Model sizes

Dataset

- Online datasets
- Entertained the idea of creating our own
- Tailored to our needs
- Started small
- Expanded with the help of a LLM
- Repeat entries
- Combining separate datasets
- Deleting repeats

Training and fine-tuning the model

- Base model chosen: CodeT5 (Initially large, changed to base)
- Dataset: combination of arithmetic custom-built, synthetic and well-known Python datasets
- Setup: training with multiple epochs and different parameters, training and validation loss monitored, progressive performance improvement
- Observation: accurate results, minor issues

Integration of complex datasets

APPS

- Broad spectrum of coding problems, from basic to advanced algorithmic tasks.

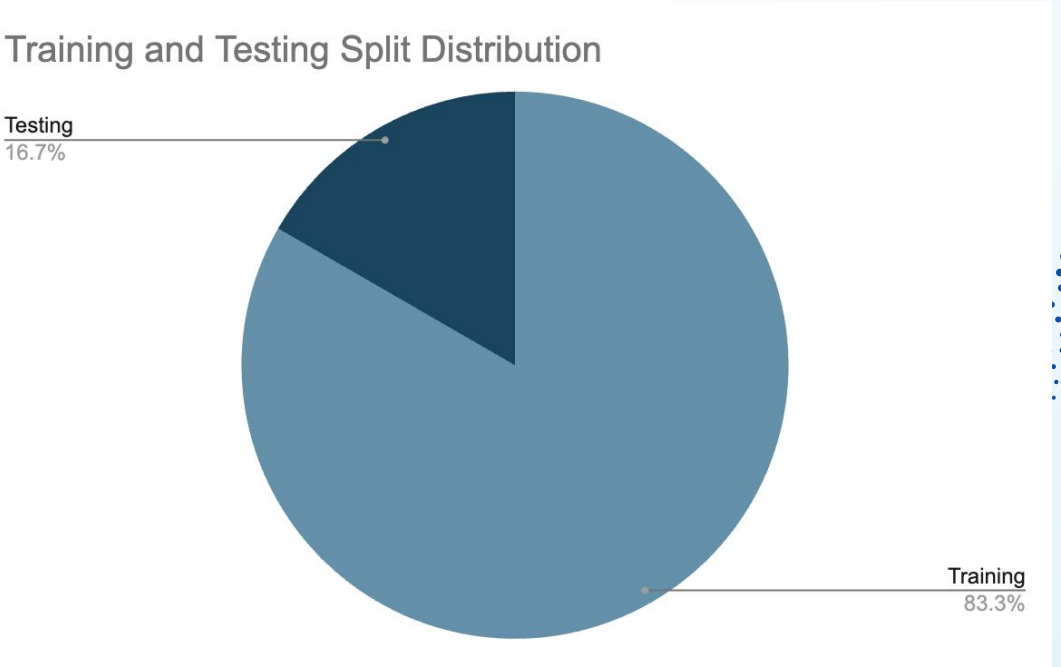
MBPP

- Logic-driven Python challenges, requiring reasoning and data manipulation.

Challenges: multiple valid solutions, different coding styles, edge cases

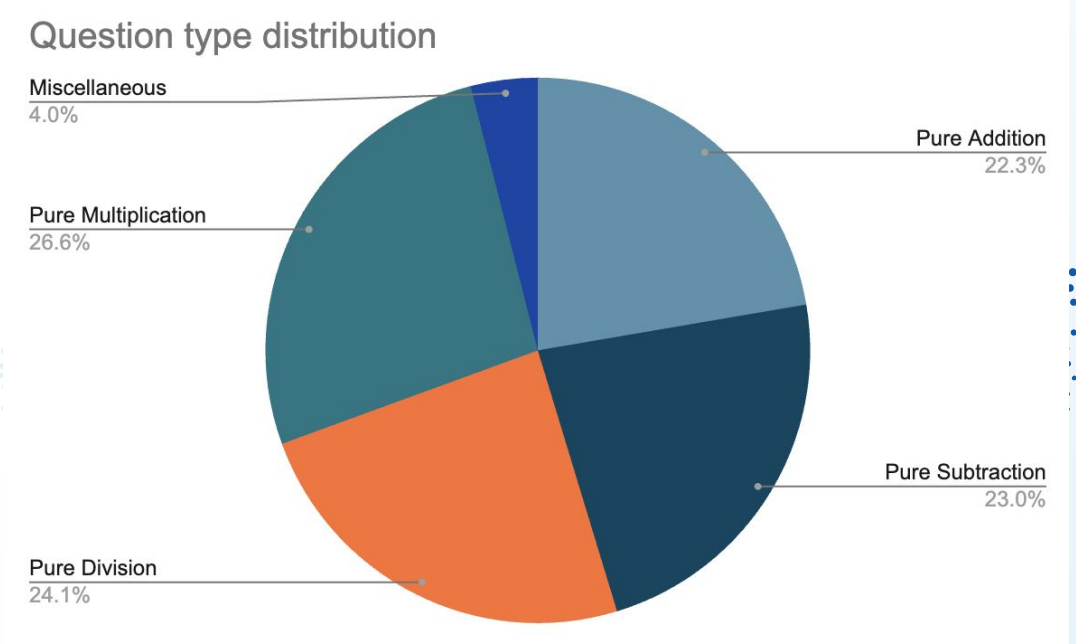
Testing and Results

- 5:1 split
  - 1,914/11,484
- Same distribution of question types as the training set



Category-Specific Distribution

- Splitting of categories:
  - *Addition, Subtraction, Multiplication, Division, and Miscellaneous*



Overall performance

- Accuracy: 99.27%
  - 1,900/1,914
- Precision: 1.000
- Recall: 0.9927
- F1 Score: 0.9963

Error analysis

- Operand misplacement
- Formatting variations
- Syntax discrepancies in complex expressions

Example I:

def solve(): return 72 / 71
def solve(): return 71 / 72

Example II:

*def solve(): return (9 ** 2 + (5 * 6 - 4 ** 3))*

*def solve(): return (9 ** 2 + 5 * 6 - 4 ** 3)*

Example III:

import math

def solve(): return math.sqrt(225)

def solve(): return sqrt(225)

Category Specific Performance

Category	Total Questions	Correct Answers	Accuracy (%)
Pure Addition	427	427	100
Pure Subtraction	440	440	100
Pure Division	462	461	99.78
Pure Multiplication	509	509	100
Miscellaneous	63	76	82.89
Overall	**1,914**	**1,901**	**99.27**

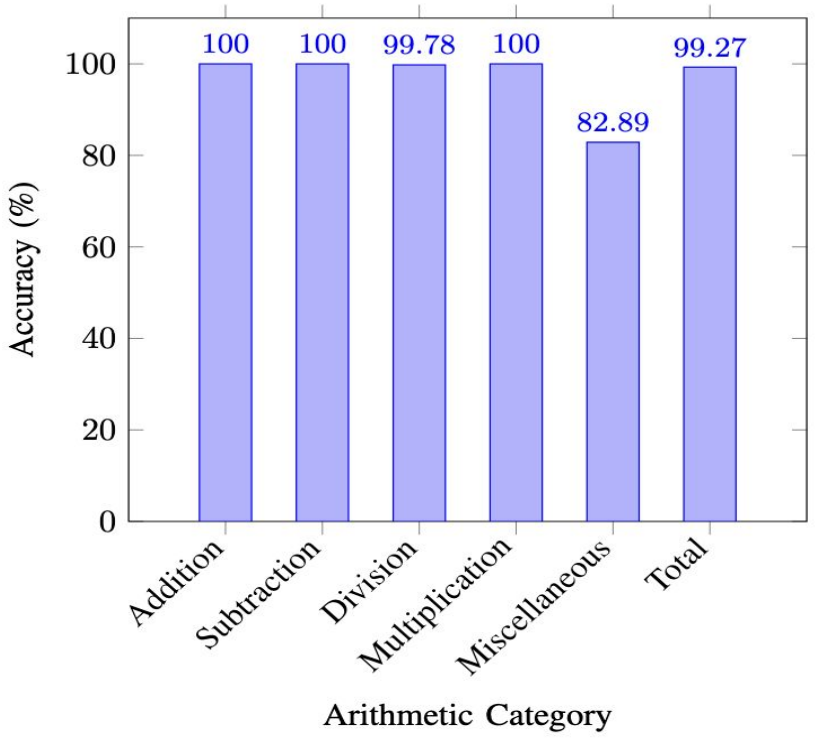


Fig. 2. Accuracy per arithmetic category.

The UI implementation

- Frontend (HTML & JS): Simple input field for math problems, syntax highlighting
- Backend (Flask & Python): Processes user input and translates it into executable Python code
- AI Model integration: fine-tuned CodeT5
- Initially execution results too - trivial for problems with variables

Math Problem Translator

Enter your math problem:

Translate to Python

Generated Python Code:

Example Problems

Math Problem Translator

Enter your math problem:

What is 8 plus 9 plus 10 plus 4

Translate to Python

Generated Python Code:

```
def solve(): return 8 + 9 + 10 + 4
```

Example Problems

- Square 5 + 9
- Add 8, the product of 6 and 9
- Take the square root of 64
- Add 4, 9 squared
- What is 8 plus 9 plus 10

Key Lessons and Conclusion

- Effectiveness of Fine-Tuning
- Error Analysis and Superficial Discrepancies
- Importance of Dataset Diversity

Q&A

If anyone has any questions, then feel free to ask!