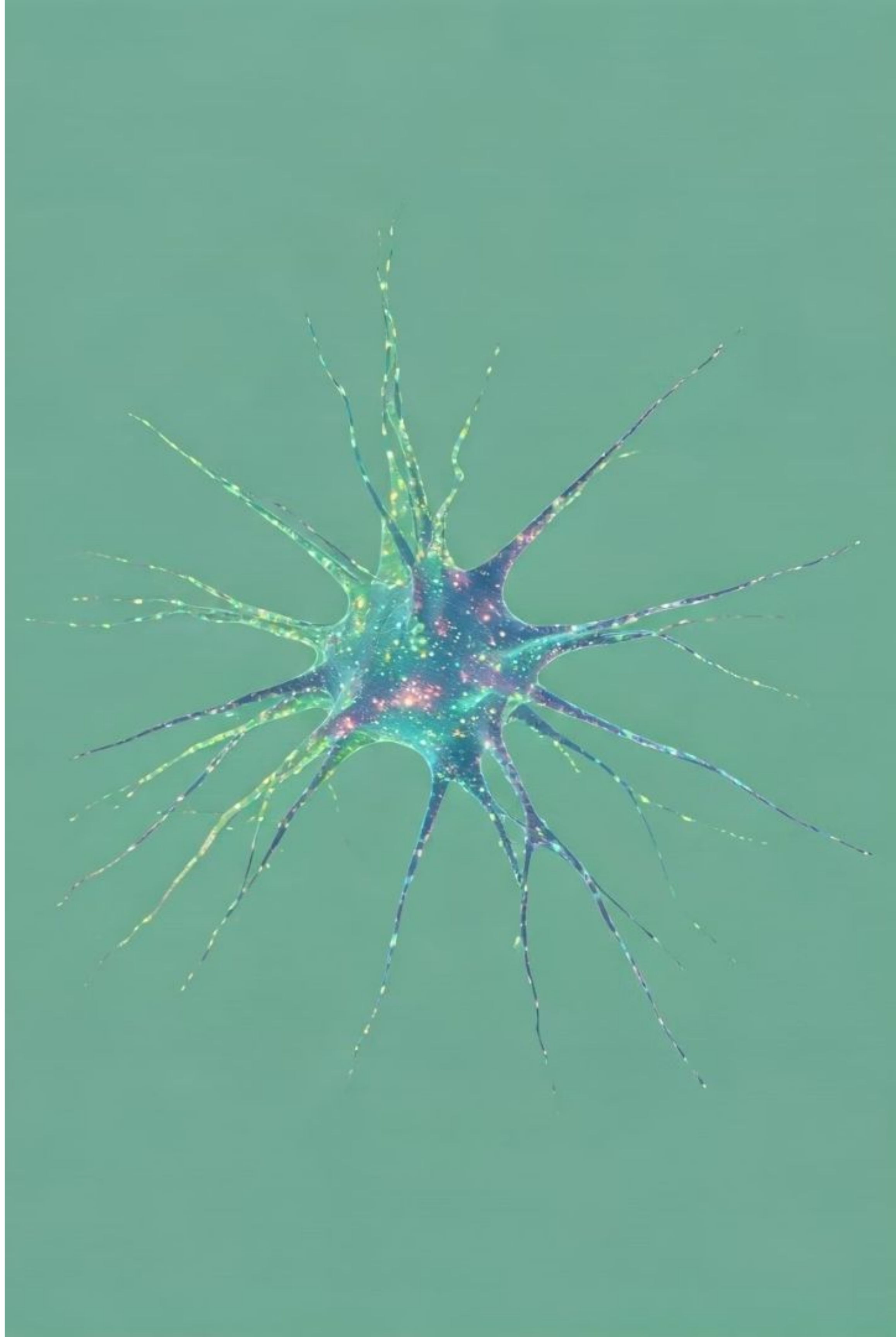
Classification of Mathematical Problems: NLP Approaches

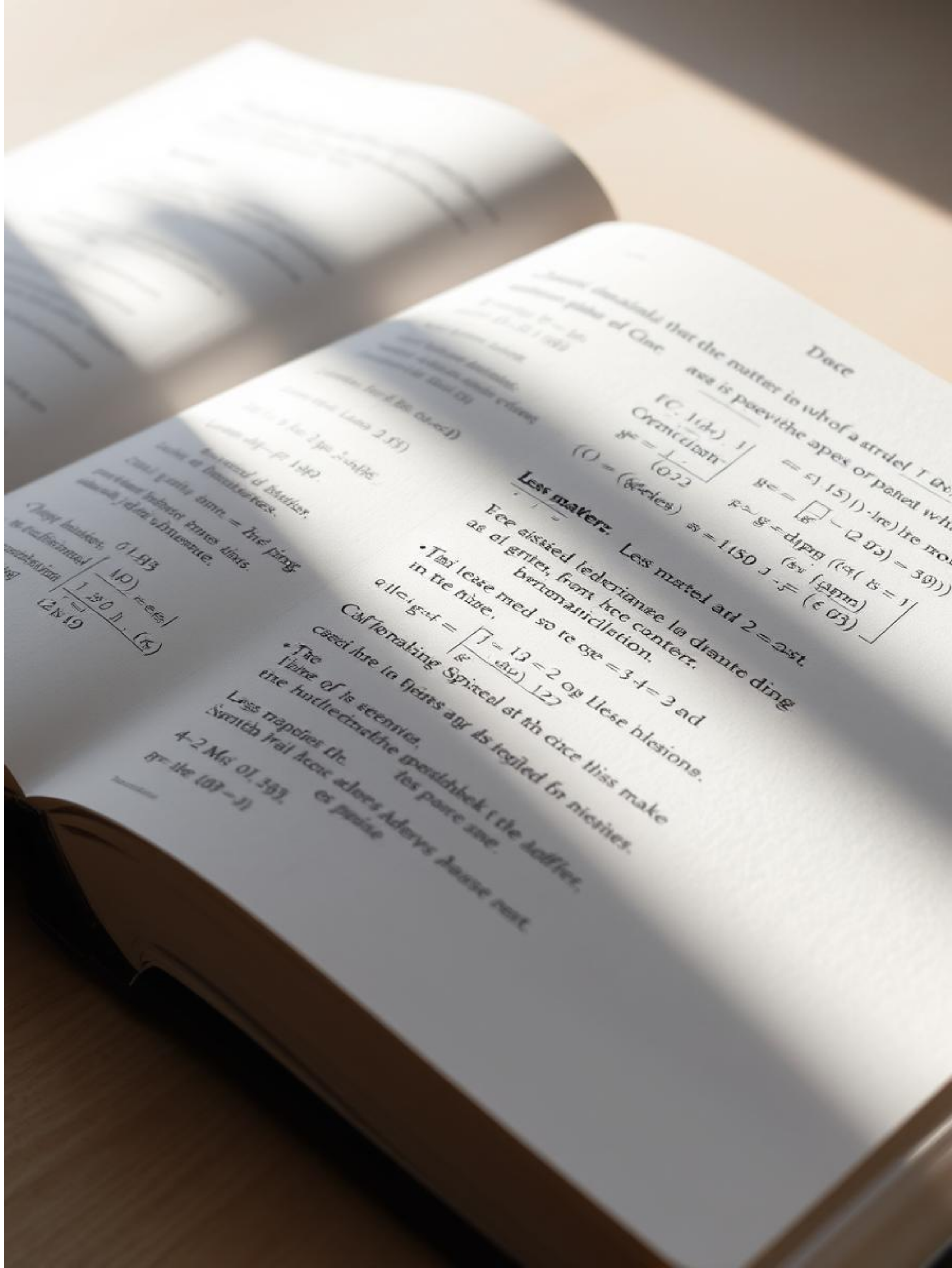
by
Deepika Reddygari
Phanindra Kumar Kalaga
Prudhviraj Chekuri



Introduction

This project explores the classification of mathematical word problems into subject areas like Algebra, Geometry, Calculus, and more using Natural Language Processing techniques. We implemented both traditional machine learning algorithms with TF-IDF features and modern transformer-based deep learning approaches.

Our work demonstrates how these techniques can aid in educational resource management, targeted learning recommendations, and organizing mathematical question repositories. The following presentation details our methodologies, experiments, and findings from this classification task.



Motivation Behind the Project

Identified Challenge

How well do these techniques adapt when the 'language' changes? Mathematical text

Inspired Solution

We envisioned building a system that understands problem language and classifies problems automatically, mimicking a teacher’s insight.

Natural Language Processing

This project explores NLP techniques to classify math problems across various subjects, improving educational resource accessibility.

Project Overview

Data Collection & Preparation

Mathematical problems dataset from Kaggle with 8 distinct categories

Model Development

Classical ML models with TF-IDF and transformer-based approaches

Evaluation & Analysis

Performance metrics, hyperparameter tuning, and overfitting prevention

Insights & Future Work

Key learnings and potential improvements

Dataset Description

Source & Structure

Dataset from Kaggle competition "Classification of Math Problems by Kasut Academy" containing training set (train.csv) with questions and category labels, test set (test.csv) for predictions.

Categories

8 distinct mathematical categories (0-7) including Algebra, Geometry, Calculus, Number Theory, Combinatorics, Probability & Statistics, Linear Algebra, and Discrete Mathematics.

Data Quality

Initial analysis revealed quality issues such as extra text and URLs, suggesting the data might have been scraped rather than manually curated. A paraphrased version (train_pp.csv) was also used for experiments. Class imbalance is also observed.

Classical NLP Models

Model Performance

Evaluation metrics on validation set

Classification Algorithms

Naive Bayes, Random Forest, LightGBM

TF-IDF Vectorization

Converting text to numerical features based on word frequency

Our classical approach began with TF-IDF vectorization, which assigns weights to words based on their frequency within a document and rarity across the corpus. We implemented this using Scikit-learn's TfidfVectorizer with various configurations.

These numerical vectors were then fed into standard classification algorithms. We found that text preprocessing significantly impacted performance, with cleaned data improving results for Logistic Regression and LightGBM models.

Transformer-Based Models

Architecture

Transformer models use self-attention mechanisms to capture contextual relationships between words. The architecture consists of an encoder to process input text and a classification head for the specific task.
The self-attention calculation is represented as:

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T/\sqrt{d_k})V$$

Models Implemented

- DistilBERT: A smaller, faster version of BERT
- DeBERTa: Incorporates disentangled attention and enhanced mask decoder
- MathBERT: Specifically pre-trained on mathematical texts from arXiv and textbooks
- T5 - Text-to-Text Transfer Transformer
- LLAMA 3.2 1B

DistilBERT

Model: distilbert-base-uncased

Parameters: ~66 Million

Backbone: DistilBERT (Distilled BERT, Encoder-Only)

Task Type: Sequence Classification

Training Setup:

- *Library* : Hugging Face Trainer
- *Epochs* : 3
- *Learning Rate* : 2×10^{-5}
- *Batch Size* : 64 (per device)
- *Optimizer* : AdamW (Weight Decay: 0.01)
- *Precision* : FP32
- *Evaluation* : Accuracy & F1 (Micro/Weighted), best model saved on F1-Micro.

DeBERTa V3

Model: microsoft/deberta-v3-base

Parameters: ~184 Million

Backbone: DeBERTa V3 (Encoder-Only)

Task Type: Sequence Classification

Training Setup:

- *Library* : Hugging Face Trainer
- *Epochs* : 10
- *Learning Rate* : 2e–5
- *Batch Size* : 8 (per device)
- *Optimizer* : AdamW (Weight Decay: 0.01)
- *Precision* : Mixed Precision (FP16)
- *Evaluation* : Accuracy & F1 (Micro/Weighted), best model saved on F1-Micro.

Score on Competition Test Set: 0.8510

KAChallenges-Deberta-V3-base - Version 1 0.8510

Complete · 6h ago · deberta-final

T5

Model: t5-base

Parameters: ~220 Million

Backbone: T5 (Encoder-Decoder)

Task Type: Sequence-to-Sequence (Text Generation of Label Name)

Training Setup:

- *Library* : Hugging Face Seq2SeqTrainer
- *Epochs* : 10
- *Learning Rate* : 5e–5
- *Batch Size* : 8 (per device)
- *Optimizer* : AdamW (Weight Decay: 0.01)
- *Input Format* : "Classify this math problem: [PROBLEM TEXT]"
- *Target Format* : "[LABEL NAME]" (e.g., "Algebra")
- *Evaluation* : Accuracy & F1 (Micro/Weighted), best model saved on F1-Micro.

Score on Competition Test Set: 0.8239

KAChallenges-T5 - Version 1

Complete · 14h ago · t5-final

0.8239

LLAMA 3.2 1B

Model: unsloth/Llama-3.2-1B

Parameters: ~1 Billion

Backbone: Llama 3.2 (Decoder-Only)

Task Type: Instruction Following (Text Generation of Label Name)

Training Setup:

- *Libraries* : unsloth, trl (SFTTrainer)
- *Technique* : LoRA (r=16), 4-bit Quantization
- *Max Steps* : 640 (~4 Epochs over ~20k samples)
- *Batch Size* : 32 (Effective Total: 4 per device * 8 grad accum)
- *Optimizer* : 8-bit AdamW, *Learning Rate* : 2e-4
- *Input Format* : Prompt (### Instruction: ... ### Input: ... ### Response: ...)
- *Target Format* : "[LABEL NAME]<|end_of_text|>" (within Response section)
- *Evaluation* : No validation during training; manual accuracy check on hold-out set post-training.

Score on Competition Test Set: 0.8346

submission.csv 0.8346

Complete · 1d ago · llama-1b-fine-tuned-851

Ensemble

Component Models:

- DeBERTa V3 Base (microsoft/deberta-v3-base) - Sequence Classification
- T5 Base (t5-base) - Sequence-to-Sequence (Label Generation)
- Llama 3.2 1B (unsloth/Llama-3.2-1B) - Instruction Following (Label Generation)

Ensemble Technique: Hard Voting

Score on Competition Test Set:

KAChallenges-Ensemble - Version 1
Complete · 4h ago · ensemble-final
0.8588

Leaderboard Rank: 15 / 180

15	**Prudhviraju Chekuri**	0.8588	27	4h

Your Best Entry!
Your most recent submission scored 0.8588, which is an improvement of your previous score of 0.8549. Great job!

Tweet this

MathBERT

Model: tbs17/MathBERT

Parameters: ~110 million

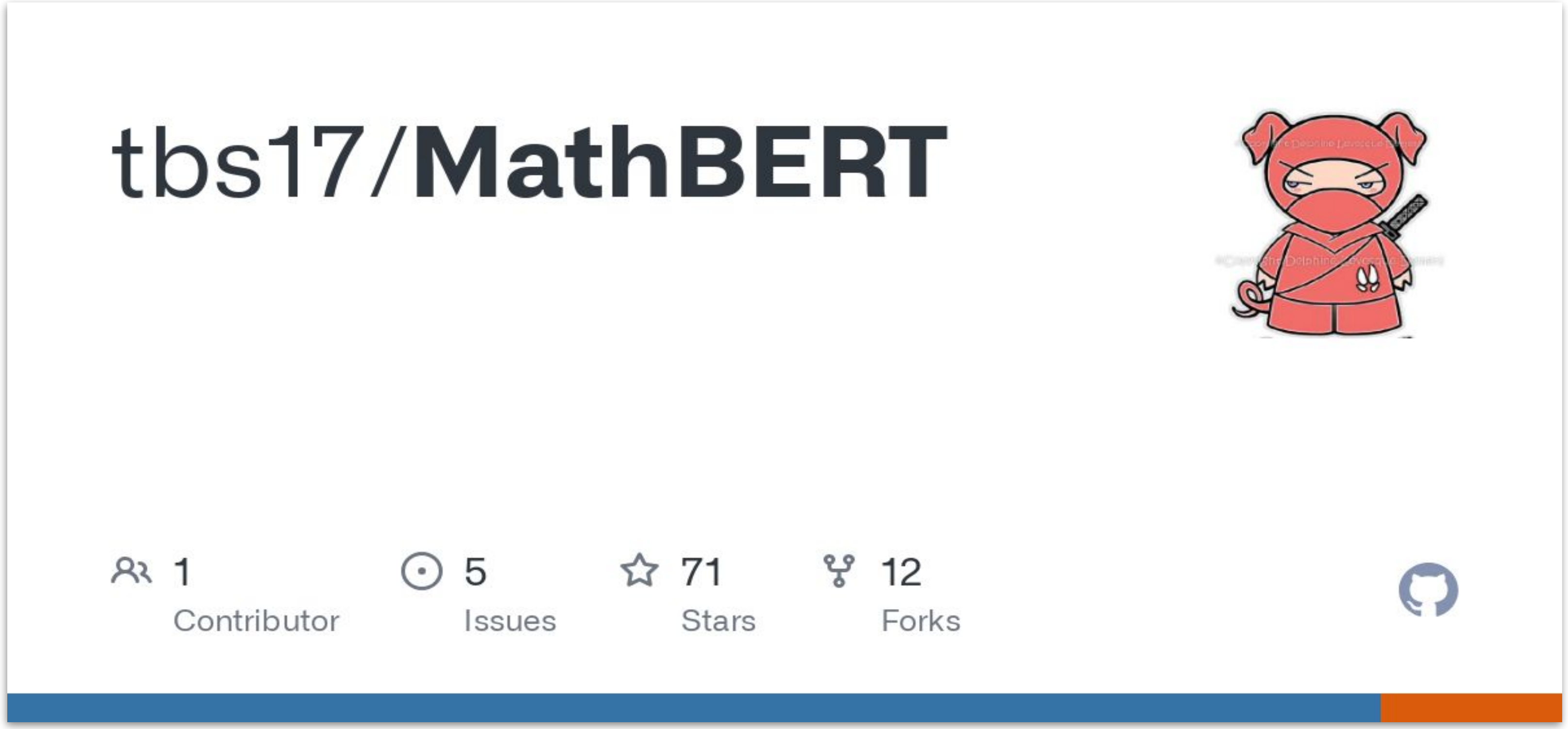
Backbone: bert-base-uncased

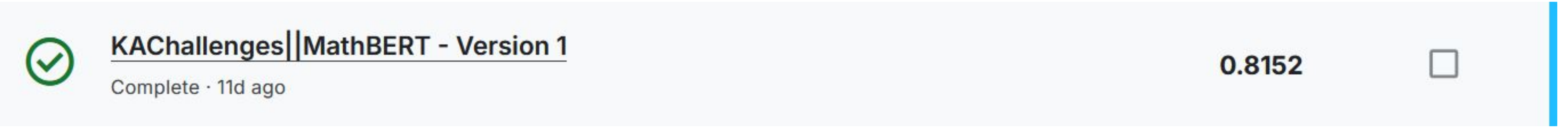
Task Type: Sequence Classification

Training Setup:

- *Library* : Hugging Face Trainer
- *Epochs* : 5
- *Learning Rate* : 2×10^{-5}
- *Batch Size* : 32
- *Optimizer* : AdamW (Weight Decay: 0.01)
- *Precision* : FP32
- *Evaluation* : Accuracy & F1 (Micro/Weighted), best model saved on F1-Micro.

Score on Competition Test Set: 0.8152





Experimental Setup

Data Splitting

80% training, 10% validation, 10% test with stratification to maintain class proportions

Implementation Frameworks

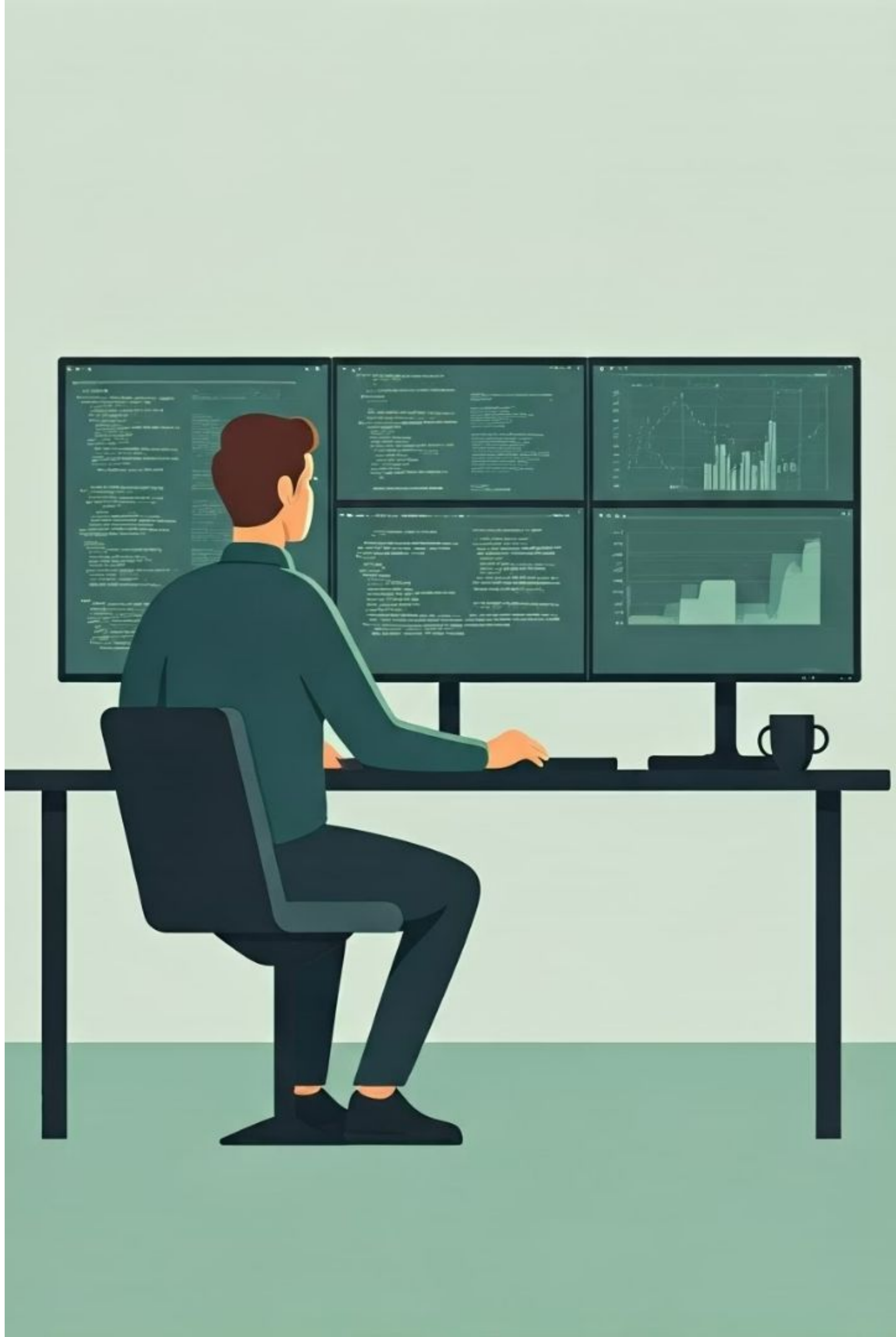
Scikit-learn for classical models; Hugging Face transformers and PyTorch for deep learning models

Performance Evaluation

Accuracy, F1-Score (macro, weighted, micro), and per-class metrics

Demonstration

Streamlit application for real-time classification of user-provided questions



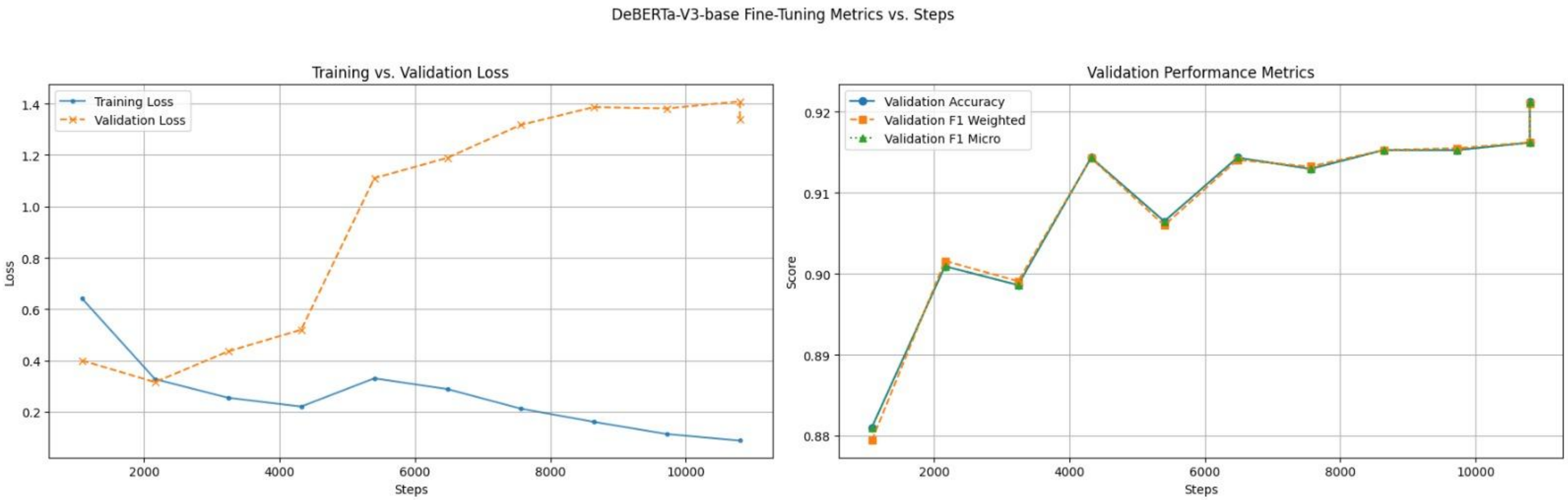
Results

Model comparison

Model Name	Training	F1-Micro (Training)	Kaggle Submission (private test)
Naive Bayes (MultinomialNB)	1.04 **seconds**	0.72	0.7040
Random Forest Classifier	6.51 **seconds**	0.74	0.7388
XGBoost	10.02 **seconds**	0.79	0.7678
LightGBM	6.24 **seconds**	0.789	0.7862
MathBERT	50 Mins.	0.83	0.8152
Llama-1b	1 Hr. 30Mins	0.85	0.8346
T5	2 Hrs.	0.83	0.8239
DeBERTa	4 Hrs.	0.9	0.851
Ensemble (Llama + T5 + DeBERTa)	-Hard Voting-	NA	0.8588

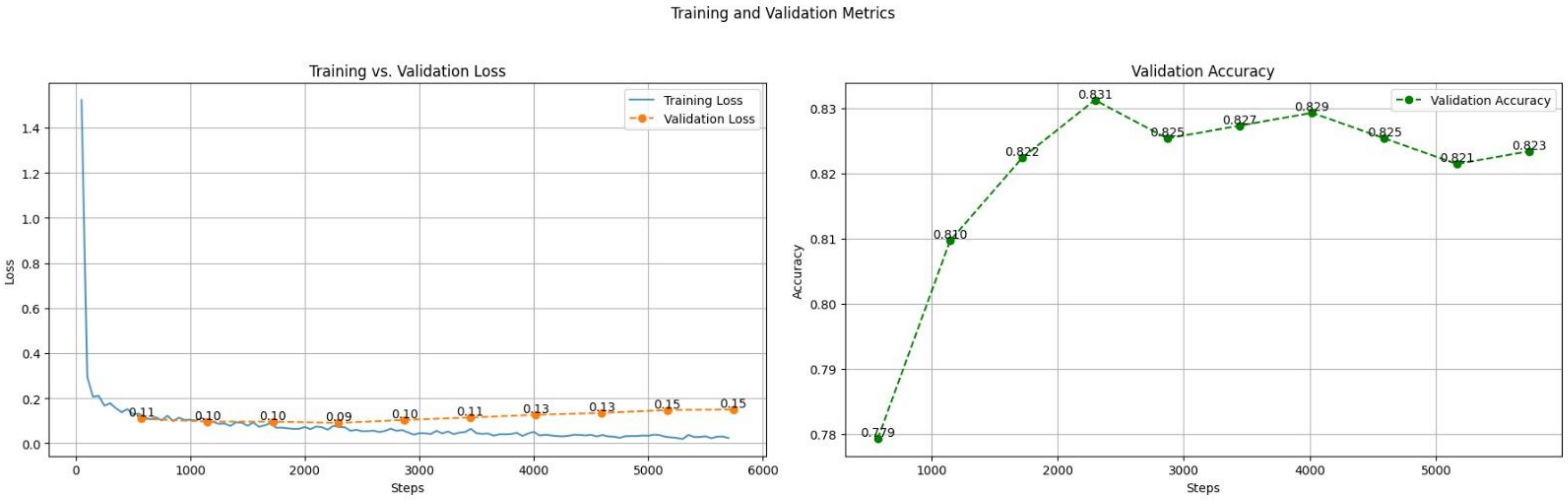
Results

Training Analysis for DeBERTa



Results

Training Analysis for T5



Streamlit Demo

Key Learnings

Model Effectiveness

Both classical and transformer-based models can be effectively applied to math problem classification

Preprocessing Importance

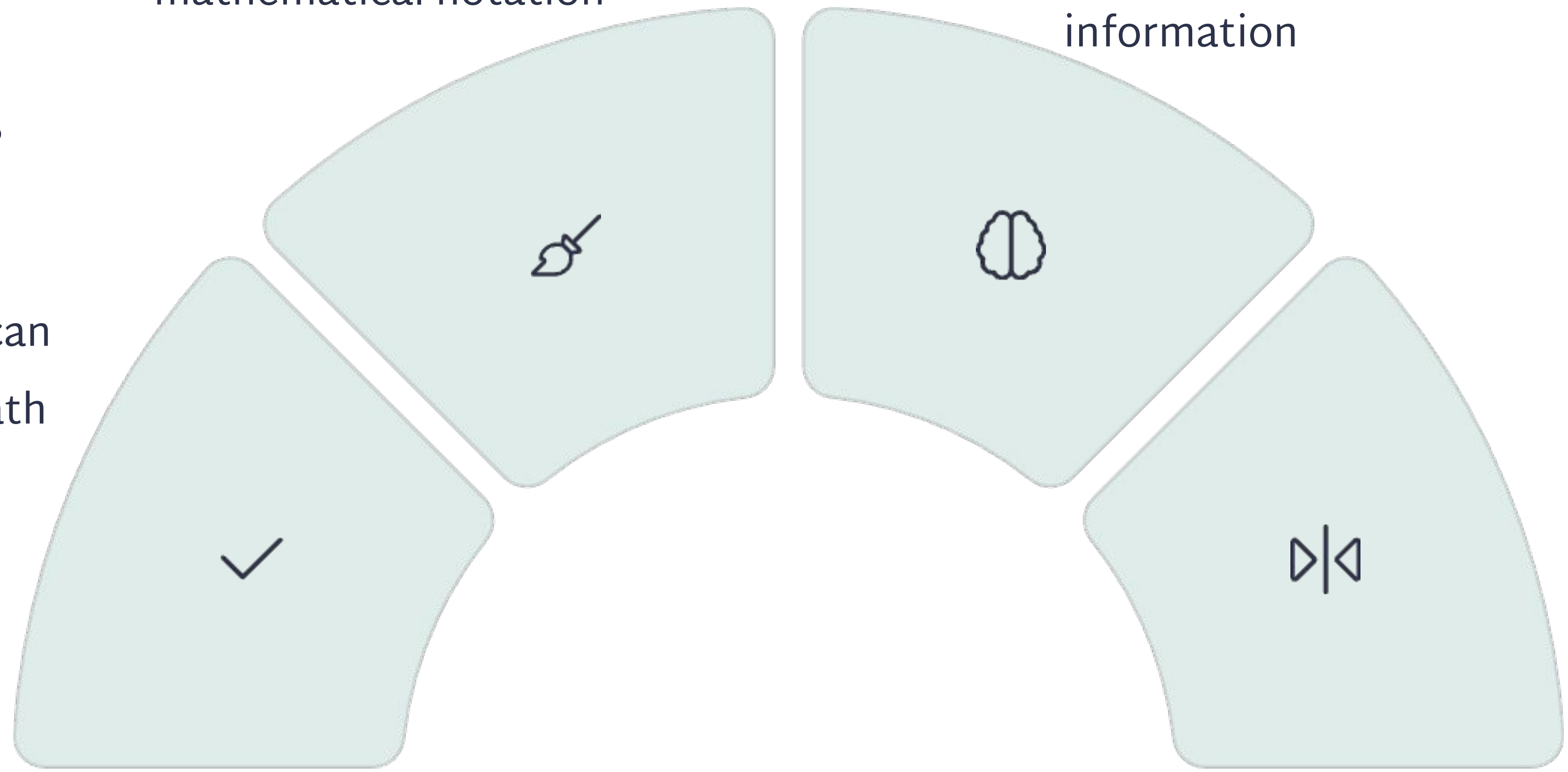
Text preprocessing is critical, especially for handling mathematical notation

Transformer Power

Transformer models, especially domain-specific ones like MathBERT, excel at capturing complex contextual information

Framework Robustness

Hugging Face library provides a robust framework for implementing and evaluating transformer models



Future Work

Hyperparameter Optimization

Experiment techniques like grid search, random search, or Bayesian optimization to further improve model performance.

*Explore possibilities of integration

Integrate this into MCP server and Agenetic Workflows to improve processing of large mathematical datasets.

Advanced Data Augmentation

Explore techniques beyond paraphrasing, such as back-translation or synonym replacement, to enhance model robustness and generalization ability.

Error Analysis & Mathematical Notation

Perform detailed analysis of model errors to identify difficult problem types and investigate sophisticated methods for representing mathematical expressions beyond simple placeholders.

Thank you!