MVP - FOCUS AI

Extension that will help you stay focused!

Szymon Smagowski, Jerzy Kraszewski

Table of Contents

INTRODUCTION

Dataset: (HTML of webpages)

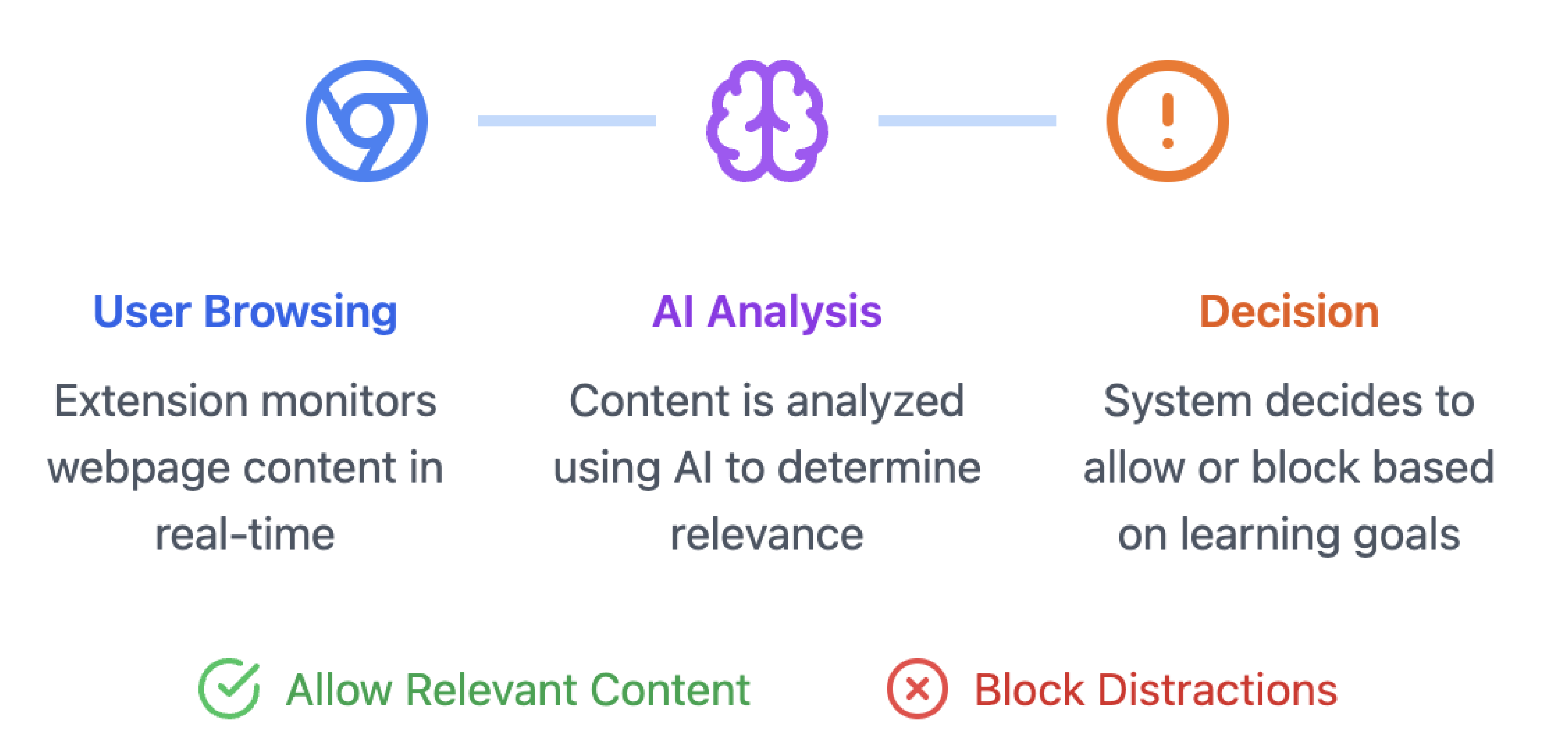
https://www.kaggle.com/datasets/uciml/identifying-interesting-web-pages/data

Content Focus Assistant

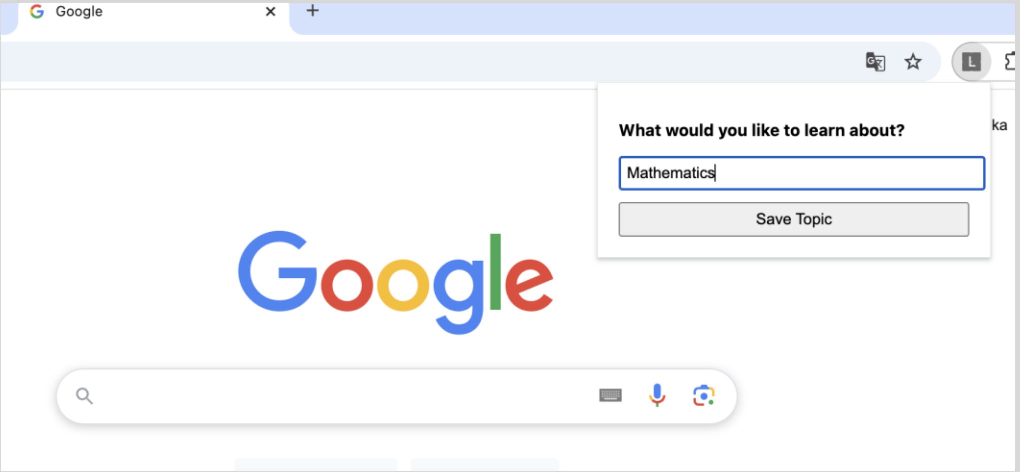
A smart application that helps users stay focused on their learning goals:

- Automatically classifies webpage content
- Prevents distractions during work or study
- Analyzes content relevance to user's objectives
- Built using LangChain + OpenAI for intelligent content analysis
- Helps users make better decisions about their time

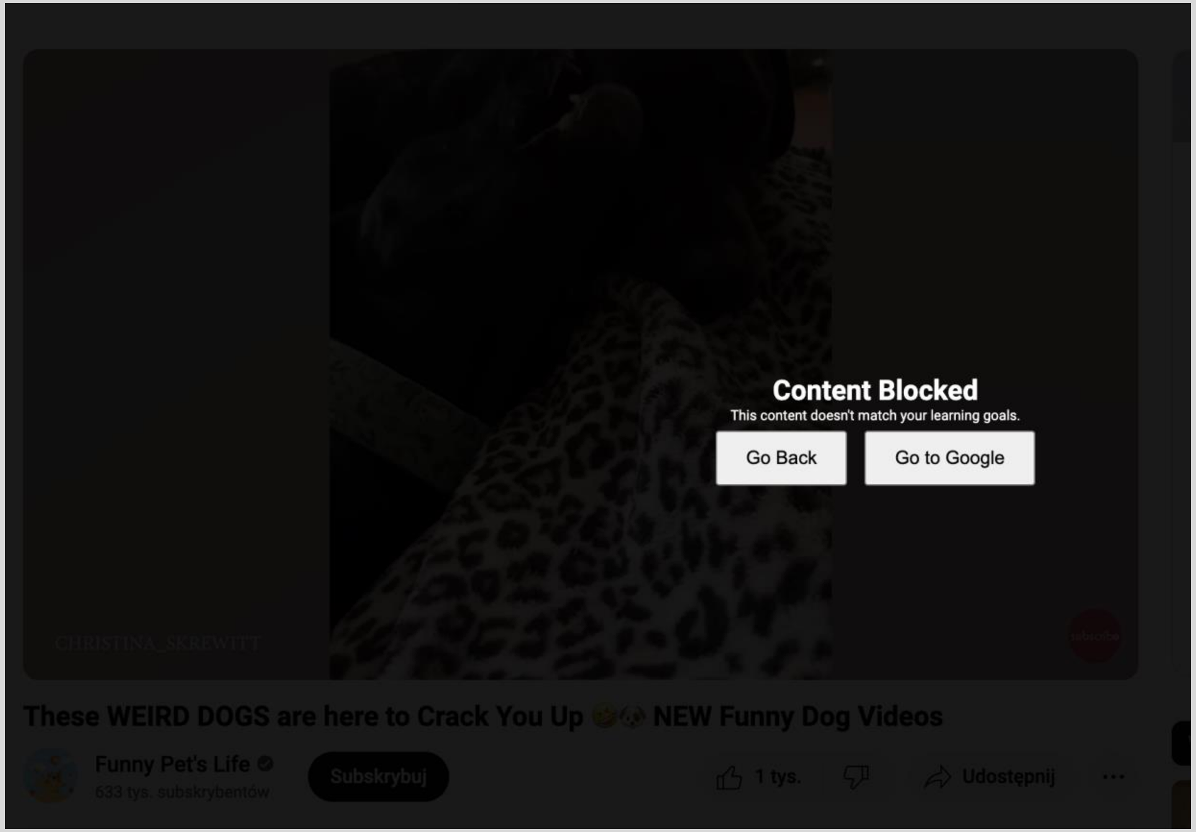
How does it work?



How does it work?



Step 1 – user decides on topic of learning



Step 2 – prompted LLM decides if content of the page is relevant for him (in this case it was blocked)

DATA PROCESSING

Dataset Overview

Kaggle Dataset

Content Categories

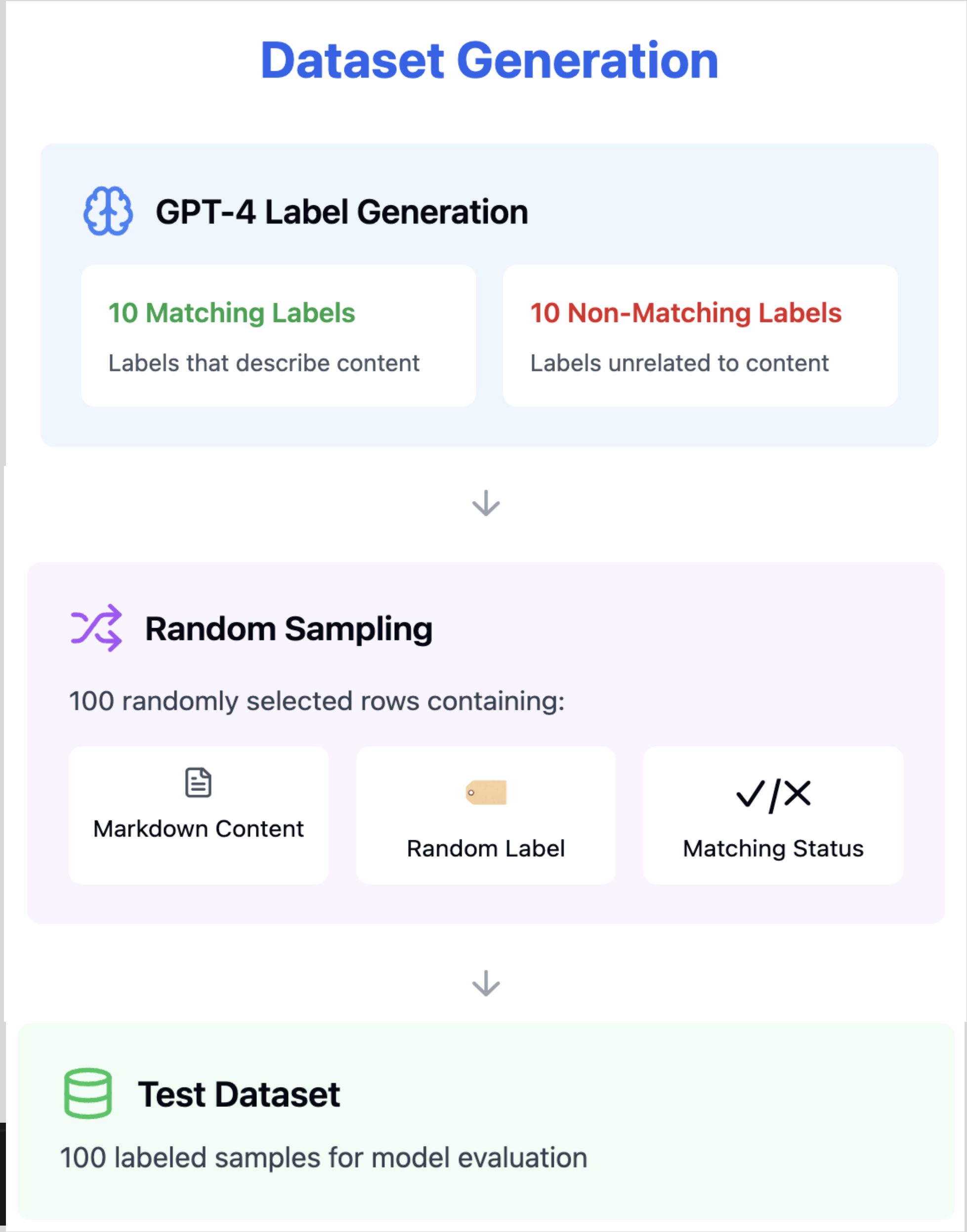
- Medical Articles
- Animal Information
- Scientific Content
- Music
- Others

Dataset Details

- ~200 HTML Pages
- Various Topics

HTML Files → Markdown Format

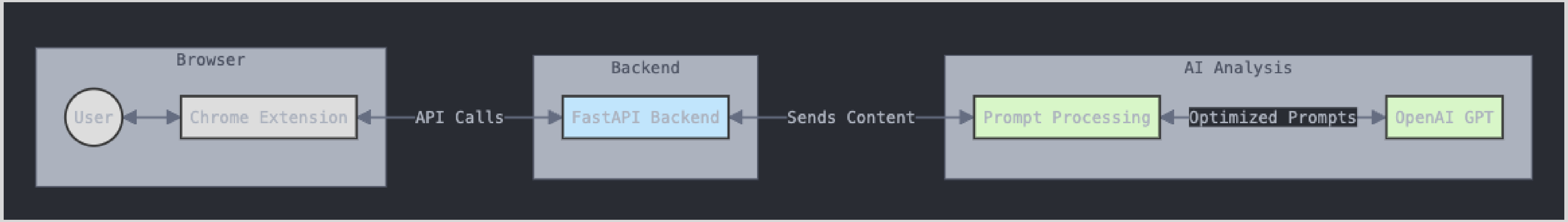
Data Processing 'Critic LLM'



NLP-Focus-AI > processed-data > ready_dataset.csv

```
markdown_content,matching_label,non_matching_label,matching_status
```

Technical implementation



- Browser section: User interaction with Chrome Extension
- Backend: FastAPI server
- AI : prompt processing and LLM decision

Prompt engineering

Prompt Engineering

Chain of Thought Example

Step-by-Step Analysis:

1. Understand the user's learning objective: '{label}'
2. Examine the page content: {content}
3. Identify key educational concepts in the content
4. Check if these concepts align with the user's learning goal
5. Consider whether this content would help achieve the learning objective
6. Evaluate if the content is worth the user's time and attention

Output:

Based on this analysis, output only 'True' if the content is relevant to the user's learning goal, or 'False' if it would be a distraction.

We included also:

- Few-shot learning
- Role prompting
- Zero-shot/Few-shot chain of thought
- Structured reasoning

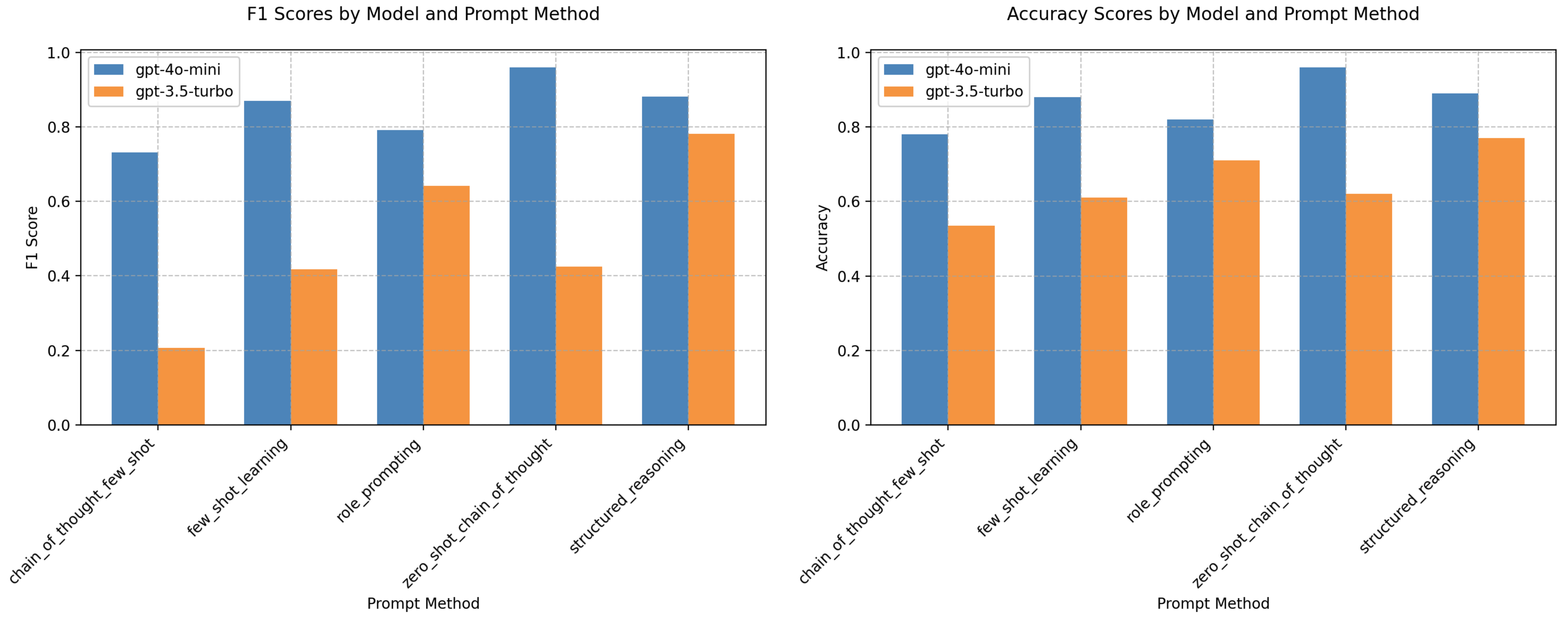
RESULTS

Binary Classification Approach

- True: Content Matches Goal
- False: Content is Distraction

Output Processing

Response validation: Extract "True" or "False" substring from LLM output


F1 Scores by Model and Prompt Method
F1 Score
gpt-4o-mini
gpt-3.5-turbo
1.0
0.8
0.6
0.4
0.2
0.0
chain_of_thought_few_shot
few_shot_learning
role_prompting
zero_shot_chain_of_thought
structured_reasoning
Prompt Method
Accuracy Scores by Model and Prompt Method
Accuracy
gpt-4o-mini
gpt-3.5-turbo
1.0
0.8
0.6
0.4
0.2
0.0
chain_of_thought_few_shot
few_shot_learning
role_prompting
zero_shot_chain_of_thought
structured_reasoning
Prompt Method

PROMPT TUNING

What is DSPy?

DSPy is a framework for programming foundation models that allows:

- Programming with LLMs using high-level primitives
- Automatic prompt optimization
- Systematic prompt engineering through modules
- Integration with various LLM providers (OpenAI, Anthropic, etc.)

DSPy Optimizers

DSPy optimizers are tools that automatically improve prompts by:

- Learning from examples (bootstrapping)
- Testing different prompt variations
- Measuring performance using defined metrics
- Selecting the best performing prompts

Key optimizer: BootstrapFewShot - Automatically generates and tests few-shot examples

```
teleprompter = BootstrapFewShot(
    metric=self.metric,
    max_bootstrapped_demos=8,
    max_labeled_demos=8,
    max_rounds=10,
)
```

USING THE SAME DATASET

```python
@dataclass
class Input:
    label: str
    content: str

@dataclass
class Output:
    classification: str
    reasoning: str = ""

class BrainrotDataset(Dataset):
    def __init__(self, data: pd.DataFrame):
        self.data = data
    # ...

class ZeroShotCoTClassifier(dspy.Module):
    # ...
```

Code Structure - Classes

Key components in the code:

- Input/Output dataclasses - Define data structure
- BrainrotDataset - Custom dataset implementation
- ZeroShotCoTClassifier - Main classifier using chain-of-thought
- BrainrotOptimizer - Manages optimization process

CLASS STRUCTURE

ZeroShotCoTClassifier Implementation

The classifier uses step-by-step reasoning:

1. Analyzes user's learning goal

2. Evaluates content relevance

3. Assesses potential distractions

4. Makes binary classification (True/False)

Uses dspy.Predict for structured output generation

```
class ZeroShotCoTClassifier(dspy.Module):
    def __init__(self):
        super().__init__()
        self.predictor = dspy.Predict("""Evaluate learning opportunity
        Return True if content supports learning, False if it's a distr

    def forward(self, label: str, content: str) -> Output:
        steps = f"""Let's evaluate:
        1. What is the user trying to learn? ({label})
        2. What knowledge does this content provide?
        3. Would it advance the user's goal?
        4. Could it distract from the objective?
        5. Is this the right time to engage?"""
```

PROMPT USED

```python
# Data splitting
train_val_size = int(0.8 * len(processed_data))
train_val_data = processed_data.iloc[:train_val_size]
test_data = processed_data.iloc[train_val_size:]

# Optimization
optimizer = BrainrotOptimizer(train_data, val_data)
results = optimizer.optimize_classifiers()

# Testing
best_name, best_classifier = optimizer.get_best_classifier()
# Final evaluation on test set
for _, row in test_data.iterrows():
    test_input = {
        'label': row['label'],
        'content': row['content']
    }
    prediction = best_classifier(**test_input)
```

Optimization Process

The optimization workflow:

1. Data split into train/validation/test sets

2. BootstrapFewShot configured with parameters

3. Classifier compilation and optimization

4. Performance evaluation on validation set

5. Final testing on held-out test set

SIMILARLY TO TRADITIONAL MODEL TRAINING

INSIGHTS

DSPy Optimizer Internals

Under the hood, DSPy optimizers work through:

• LLM-driven Bootstrapping: Uses a 'critic' LLM (typically GPT-4) to evaluate and improve prompts

• Iterative Refinement: Conducts multiple rounds of testing and improvement (usually 5-10 rounds)

• Example Generation: Automatically creates new few-shot examples by analyzing successful and failed cases

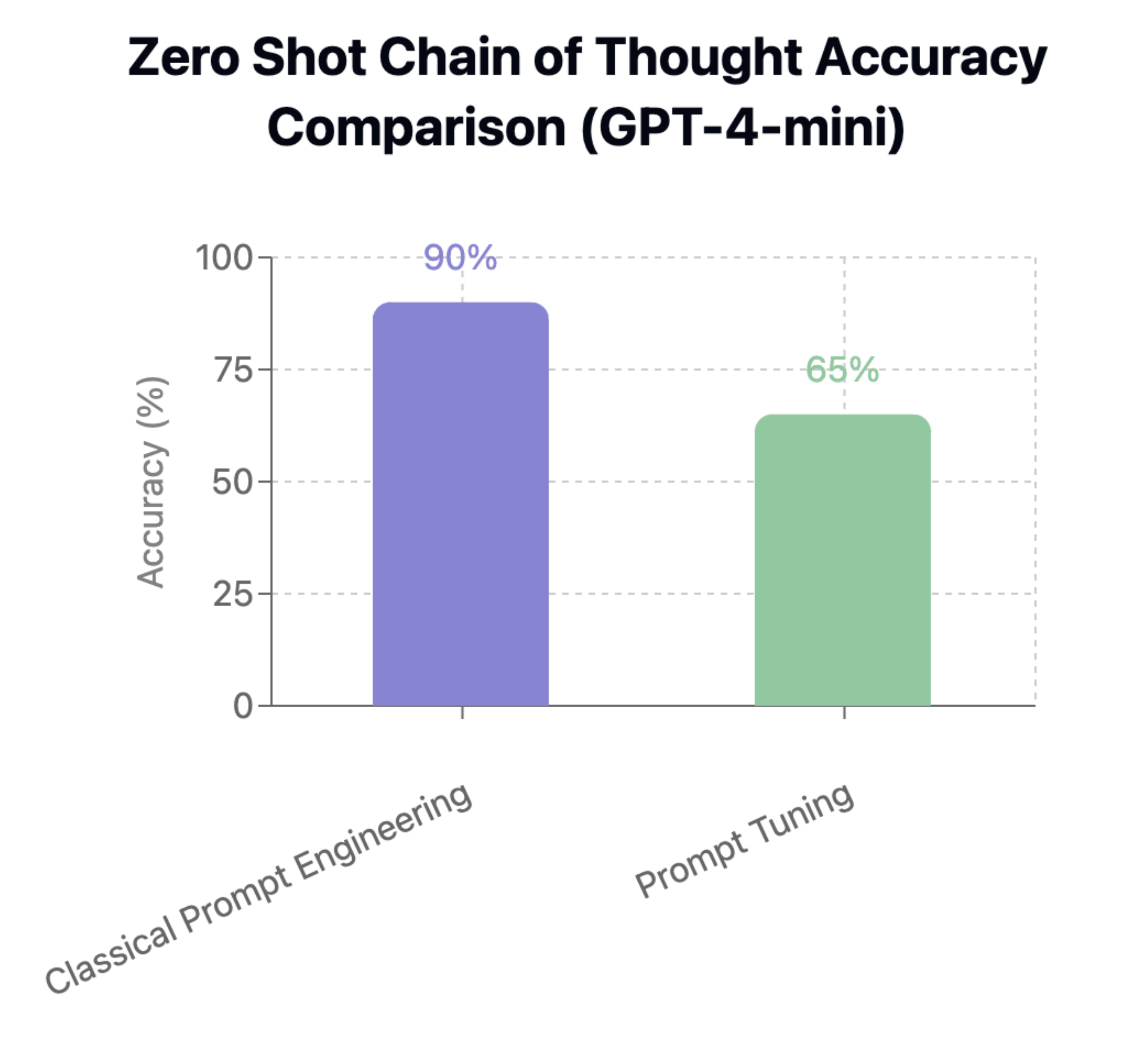
• Prompt Evolution: Modifies instruction templates based on performance feedback

• Performance Tracking: Measures improvements using customizable metrics like accuracy or F1 score

RESULTS

Why is it worse?

- Exact match for the output, instead of regex 'True/False'
- Library limitation in terms of compatibility of versions and dependencies, old documentation
- Too small dataset



LangChain Ecosystem

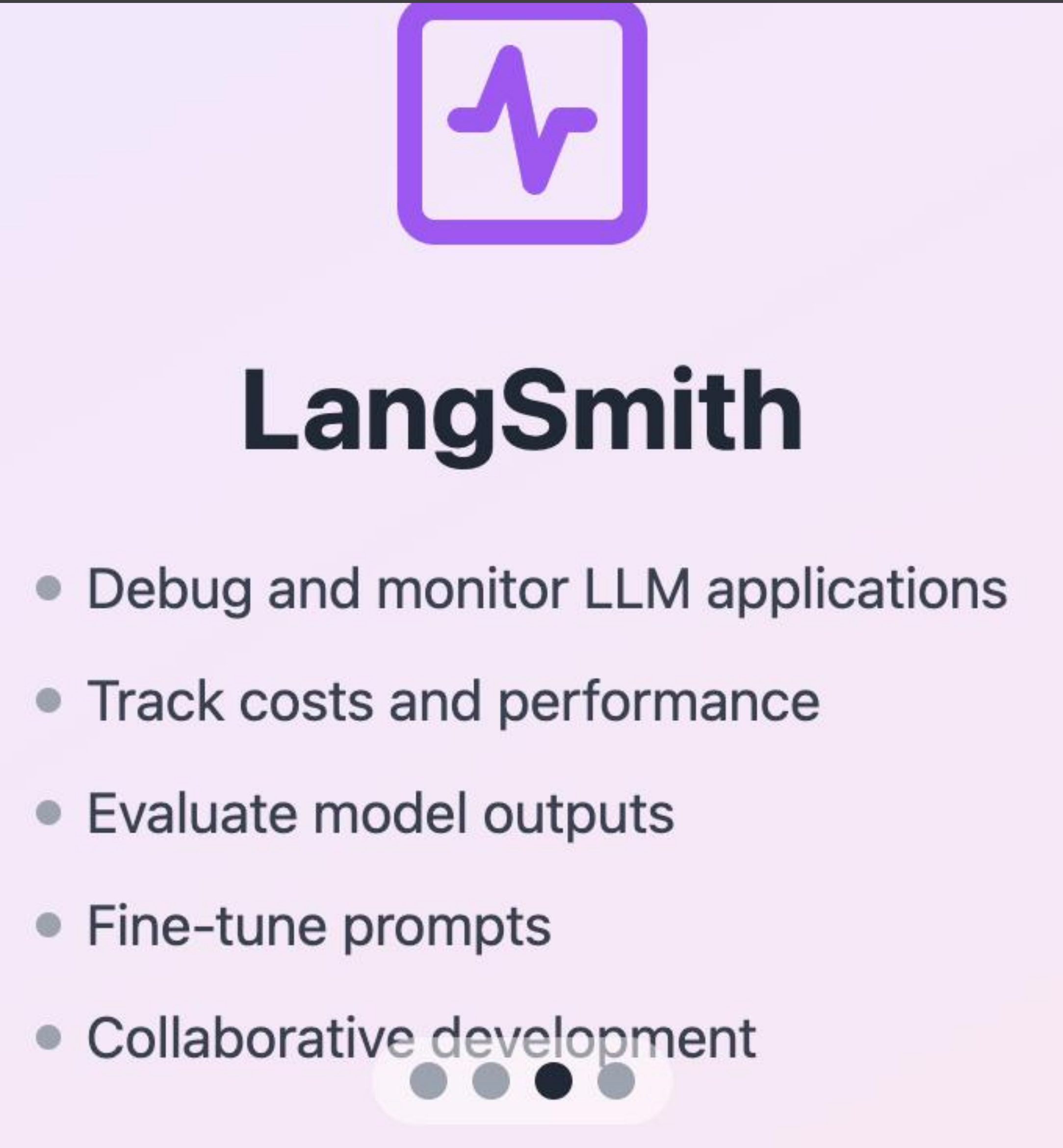
Powerful tools for building AI applications

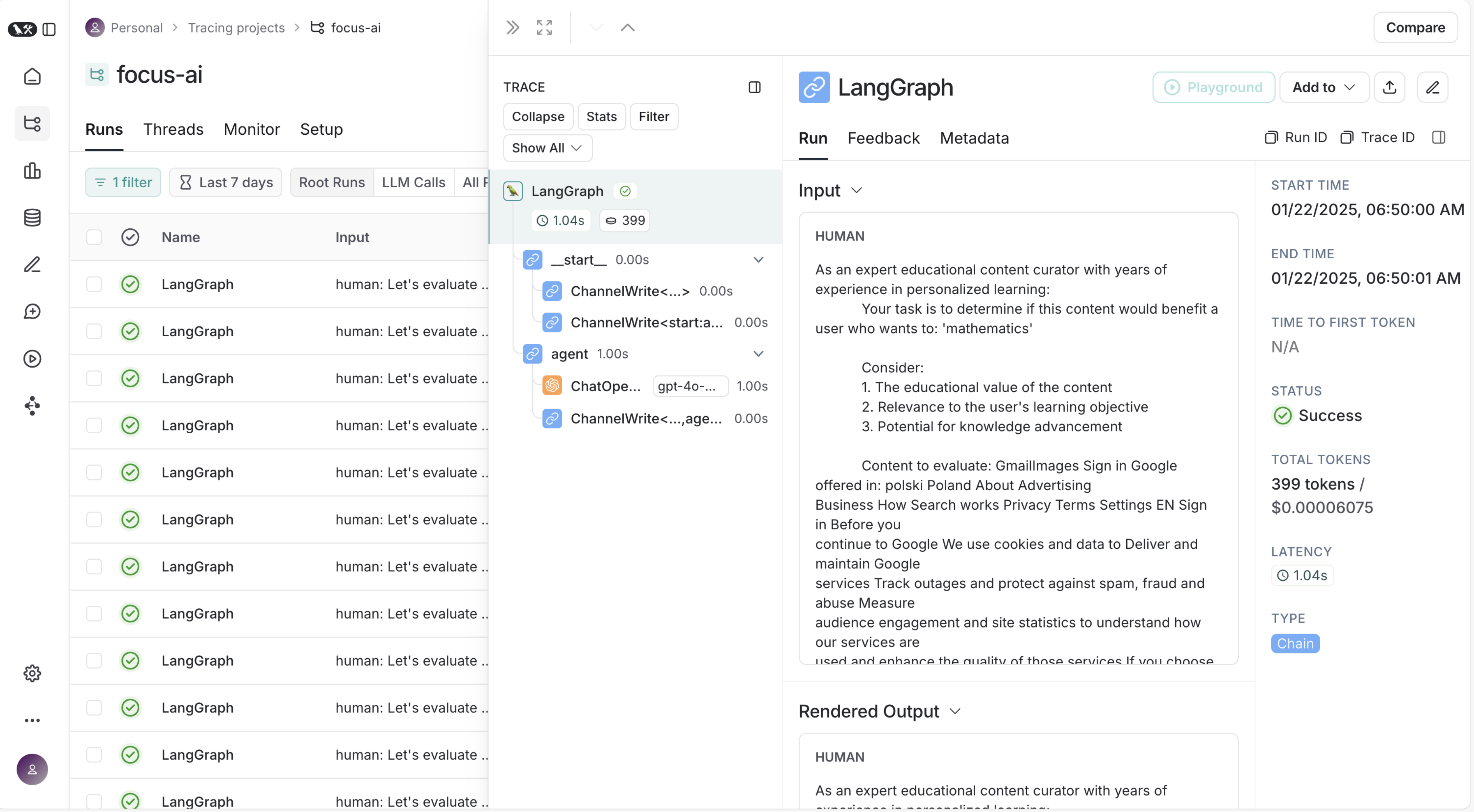
```
self.model = ChatOpenAI(
    model_name=settings.OPENAI_MODEL,
    api_key=settings.OPENAI_API_KEY
)
```

LangChain

- Framework for developing LLM applications
- Chains and prompts management
- Multiple LLM providers support
- Built-in memory systems
- Powerful agents and tools

```python
os.environ["LANGCHAIN_API_KEY"] = settings.LANGCHAIN_API_KEY
os.environ["LANGCHAIN_TRACING_V2"] = settings.LANGCHAIN_TRACING_V2
os.environ["LANGCHAIN_PROJECT"] = settings.LANGCHAIN_PROJECT
```




Personal > Tracing projects > focus-ai
focus-ai
Runs
Threads
Monitor
Setup
1 filter
Last 7 days
Root Runs
LLM Calls
All
Name
Input
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
LangGraph
human: Let's evaluate ..
Compare
TRACE
Collapse
Stats
Filter
Show All
LangGraph
1.04s
399
__start__ 0.00s
ChannelWrite<...> 0.00s
ChannelWrite<start:a... 0.00s
agent 1.00s
ChatOpe... gpt-4o-... 1.00s
ChannelWrite<...,age... 0.00s
LangGraph
Playground
Add to
Run
Feedback
Metadata
Run ID
Trace ID
Input
HUMAN
As an expert educational content curator with years of
experience in personalized learning:
Your task is to determine if this content would benefit a
user who wants to: 'mathematics'
Consider:
1. The educational value of the content
2. Relevance to the user's learning objective
3. Potential for knowledge advancement
Content to evaluate: GmailImages Sign in Google
offered in: polski Poland About Advertising
Business How Search works Privacy Terms Settings EN Sign
in Before you
continue to Google We use cookies and data to Deliver and
maintain Google
services Track outages and protect against spam, fraud and
abuse Measure
audience engagement and site statistics to understand how
our services are
used and enhance the quality of those services If you choose
Rendered Output
HUMAN
As an expert educational content curator with years of
START TIME
01/22/2025, 06:50:00 AM
END TIME
01/22/2025, 06:50:01 AM
TIME TO FIRST TOKEN
N/A
STATUS
Success
TOTAL TOKENS
399 tokens /
$0.00006075
LATENCY
1.04s
TYPE
Chain

```python
def get_workflow(self):
    # Create workflow graph
    workflow = StateGraph(MessagesState)

    # Define nodes
    workflow.add_node("agent", self.call_model)

    # Set entry point
    workflow.add_edge(START, "agent")

    workflow.add_edge("agent", END)

    return workflow
```

LangGraph

- Build complex AI workflows
- Graph-based orchestration
- State management
- Parallel execution
- Advanced routing capabilities

FOCUS AI GRAPH

THANK YOU!