Project: Embedding-Based Reasoning on Word Problems

This project uses vector embeddings and retrieval to help a language model solve basic arithmetic word problems by learning from semantically similar solved examples.

This notebook demonstrates:

- Using semantic embeddings to represent math word problems
- Retrieving similar examples from a solved dataset
- Prompting an LLM with contextually relevant examples
- Evaluating accuracy of reasoning via embedding-based retrieval

Step 1: Dataset Preparation

Goal: Create a dataset of 1000 simple arithmetic word problems.

Each example is a natural-language question paired with a numeric answer:

```
Q: If Jay had 5 apples and got 4 more, how many apples does
he have now?
A: 9
```

- The dataset contains 1000 basic arithmetic questions.
- Each entry includes:
  - A natural language word problem
  - The correct answer
  - The underlying operation (addition, subtraction, etc.)

In [106…

```
import pandas as pd
file_path = "/content/basic_word_problems.csv"  # If you've uploaded it manu
df = pd.read_csv(file_path)
print("Dataset loaded successfully")
df.head()
```

```
Dataset loaded successfully
```

Out[106…

	question	answer	operation
0	If Sara had 42 oranges and got another 47 oran...	89	addition
1	Noah has 7 boxes of oranges, each containing 7...	49	multiplication
2	If Mia had 23 oranges and got another 19 orang...	42	addition
3	Jay has 12 balls and wants to divide them equa...	6	division
4	If Noah had 38 marbles and got another 45 marb...	83	addition

Step 2: Embedding the Dataset

Goal: Convert all questions into high-dimensional semantic vectors.

- Use OpenAI's `text-embedding-ada-002` or any similar embedding model
- For each question:
  - Generate an embedding vector
  - Store it along with the original question and its answer
- Save the results locally (e.g., CSV or Pickle)

We use a sentence embedding model to convert each question into a fixed-size vector representation. This allows similarity comparison between word problems using cosine distance.

In [107…

```
import os

# OpenAI API Key
os.environ["OPENAI_API_KEY"] = "sk-proj-njtXyLHeWlnvhdEJzT2LS_VHhMxE1e5Bq_gL
```

In [108…

```
import openai

# Load the key from environment
openai.api_key = os.getenv("OPENAI_API_KEY")
```

In [109…

```
!pip install openai tqdm
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.11/dist-pack
ages (1.78.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packag
es (4.67.1)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/
dist-packages (from openai) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.1
1/dist-packages (from openai) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
1/dist-packages (from openai) (0.28.1)
Requirement already satisfied: jiter<1,>=0.4.0 in /usr/local/lib/python3.11/
dist-packages (from openai) (0.9.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.
11/dist-packages (from openai) (2.11.4)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-pac
kages (from openai) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/
python3.11/dist-packages (from openai) (4.13.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-p
ackages (from anyio<5,>=3.5.0->openai) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-pac
kages (from httpx<1,>=0.23.0->openai) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/di
st-packages (from httpx<1,>=0.23.0->openai) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-p
ackages (from httpcore==1.*->httpx<1,>=0.23.0->openai) (0.16.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyth
on3.11/dist-packages (from pydantic<3,>=1.9.0->openai) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/pytho
n3.11/dist-packages (from pydantic<3,>=1.9.0->openai) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/py
thon3.11/dist-packages (from pydantic<3,>=1.9.0->openai) (0.4.0)
```

Embedding Function

This function uses OpenAI's embedding model to encode a list of word problems into high-dimensional vectors. We apply this to all training examples and test queries.

In [110…

```
import os
import time
import openai
import pandas as pd
from tqdm import tqdm

client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

df = pd.read_csv("/content/basic_word_problems.csv")

model_name = "text-embedding-ada-002"
batch_size = 100
questions = df["question"].tolist()
embeddings = []
```

```
# Embedding function with batch support
def get_batch_embeddings(batch):
    try:
        response = client.embeddings.create(
            input=batch,
            model=model_name
        )
        return [item.embedding for item in response.data]
    except Exception as e:
        print(f"⚠️ Batch failed: {e}")
        return [None] * len(batch)

print(f"🔁 Embedding {len(questions)} questions in batches of {batch_size}..
for i in tqdm(range(0, len(questions), batch_size)):
    batch = questions[i:i + batch_size]
    batch_embeddings = get_batch_embeddings(batch)
    embeddings.extend(batch_embeddings)
    time.sleep(1)  # ⏱️ Delay to avoid rate limits

df["embedding"] = embeddings
output_path = "/content/embedded_word_problems.csv"
df.to_csv(output_path, index=False)

print(f"\n✅ All embeddings saved to: {output_path}")
```

```
🔁 Embedding 800 questions in batches of 100...

100%|██████████| 8/8 [00:14<00:00,  1.81s/it]
✅ All embeddings saved to: /content/embedded_word_problems.csv
```

Step 3: Semantic Retrieval

Goal: Retrieve similar solved examples when a new question is asked.

- Embed the new query
- Calculate cosine similarity with all stored question vectors
- Retrieve the most relevant example(s) from the dataset
- Use Top-1 or Top-K similarity (e.g., top 3)

In [111…
```
import os
os.environ["OPENAI_API_KEY"] = "sk-proj-njtXyLHeWlnvhdEJzT2LS_VHhMxE1e5Bq_gL
os.environ["HF_API_KEY"] = "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"
```

In [112…
```
import openai
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Load the embedded dataset
df = pd.read_csv("/content/embedded_word_problems.csv")
df["embedding"] = df["embedding"].apply(lambda x: eval(x) if isinstance(x, s
# Set OpenAI API key
openai.api_key = os.getenv("OPENAI_API_KEY")
```

Given a new question, we find the most semantically similar example from the training set based on cosine similarity.

In [113…
```
from openai import OpenAI

# Initialize OpenAI v1 client properly
openai_client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

def embed_query(query_text):
    response = openai_client.embeddings.create(
        input=[query_text],
        model="text-embedding-ada-002"
    )
    return response.data[0].embedding
```

In [114…
```
def find_top_k_similar(query, operation_type, k=1):
    query_vector = embed_query(query)
    filtered_df = df[df["operation"] == operation_type].copy()

    all_vectors = np.array(filtered_df["embedding"].tolist())
    similarities = cosine_similarity([query_vector], all_vectors)[0]
    top_k_indices = similarities.argsort()[-k:][::-1]

    results = filtered_df.iloc[top_k_indices].copy()
    results["similarity"] = similarities[top_k_indices]
    return results
```

In [115…
```
import os
from huggingface_hub import InferenceClient

HF_API_KEY = os.environ.get("HF_API_KEY")

if HF_API_KEY:
    print("✅ Token loaded")
else:
    print("❌ Token not found")
```

```
✅ Token loaded
```

🧠 Step 4: Reasoning with Retrieved Example (LLM Option)

Goal: Use a language model to answer a new question based on a similar solved example.

- Format the prompt with both:
  - A retrieved example (solved)
  - The new input question (unsolved)

Example prompt:

```
Q: Emma had 3 candies. She got 5 more. How many now?
A: 8

Q: Jay had 4 candies. He got 7 more. How many now?
A:
```

- The model will learn from the pattern and complete the new answer
- This approach leverages in-context learning with no additional training

We construct a prompt using the most similar solved example and append the new question to it. This forms an in-context few-shot prompt for the LLM.

In [116…
```python
# ✅ Load the embedded dataset from Step 3
import pandas as pd
import numpy as np

df_embedded = pd.read_csv("/content/embedded_word_problems.csv")

# ✅ Convert embedding column from string to list
df_embedded["embedding"] = df_embedded["embedding"].apply(eval).apply(np.arr
```

In [117…
```python
# ✅ Use same OpenAI embedding model and client as Step 3
import openai
import os

client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

def embed_question_openai(text, model="text-embedding-ada-002"):
    try:
        response = client.embeddings.create(
            input=[text],
            model=model
        )
        return np.array(response.data[0].embedding)
    except Exception as e:
        print("❌ Embedding Error:", e)
        return None

# 📝 Define your new question
new_question = "If Lucas had 8 marbles and found 5 more, how many marbles do
new_embedding = embed_question_openai(new_question)
```

In [118…
```python
from sklearn.metrics.pairwise import cosine_similarity

# ✏️ Specify the operation type for the new question
operation_type = "addition"  # e.g., "addition", "subtraction", "multiplicat

# 🔍 Filter dataset by operation
df_filtered = df_embedded[df_embedded["operation"] == operation_type].copy()

# 🧠 Compute similarity only within filtered set
```

```
df_filtered["similarity"] = df_filtered["embedding"].apply(
    lambda x: cosine_similarity([new_embedding], [x])[0][0]
)

# 🔝 Get the most similar match within the operation group
top_match = df_filtered.sort_values("similarity", ascending=False).iloc[0]
retrieved_question = top_match["question"]
retrieved_answer = top_match["answer"]

# 🔎 Preview the match
print(f"🔁 Top Similar Solved Example (Operation: {operation_type}):")
print("Q:", retrieved_question)
print("A:", retrieved_answer)
```

```
🔁 Top Similar Solved Example (Operation: addition):
Q: If Noah had 8 marbles and got another 34 marbles, how many marbles does N
oah have now?
A: 42
```

In [119…

```
# 🧱 Use top match + new question to create a prompt
final_prompt = f"""Here is a solved example:

Q: {retrieved_question}
A: {retrieved_answer}

Now solve this similar problem:

Q: {new_question}
A:"""

print("\n📤 Prompt Sent to Model:\n")
print(final_prompt)
```

```
📤 Prompt Sent to Model:

Here is a solved example:

Q: If Noah had 8 marbles and got another 34 marbles, how many marbles does N
oah have now?
A: 42

Now solve this similar problem:

Q: If Lucas had 8 marbles and found 5 more, how many marbles does he have no
w?
A:
```

In [120…

```
# 🤖 Query the Together AI-hosted Llama-4-Scout-17B model
from huggingface_hub import InferenceClient
import time

# ✅ Set up inference client
llama_client = InferenceClient(
    provider="together",
    api_key=os.getenv("HF_API_KEY")
)
```

```
def query_llama4(prompt, max_tokens=256, temperature=0.7):
    try:
        start = time.time()
        result = llama_client.chat.completions.create(
            model="meta-llama/Llama-4-Scout-17B-16E-Instruct",
            messages=[{"role": "user", "content": prompt}],
            max_tokens=max_tokens,
            temperature=temperature
        )
        elapsed = time.time() - start
        return result.choices[0].message.content.strip(), elapsed
    except Exception as e:
        print("❌ API Error:", e)
        return None, None

# 🧪 Run the prompt
response, elapsed = query_llama4(final_prompt)

# 📤 Show the result
if response:
    print(f"\n✅ Model Response ({round(elapsed, 2)} sec):\n{response}")
else:
    print("⚠️ No response received.")
```

```
✅ Model Response (0.9 sec):
To solve this problem, I'll follow the same steps as the example.

Lucas had 8 marbles and found 5 more. To find the total number of marbles he
has now, I'll add 8 and 5.

8 + 5 = 13

So, Lucas has 13 marbles now.

A: 13
```

In [120…

In [120…

📈 Step 5: Evaluation & Visualization

Goal: Evaluate model accuracy and performance.

- Test on a held-out set of word problems
- Compare model's predicted answers vs. ground truth
- Track:
    - Accuracy per arithmetic operation
    - Inference time per question
- Visualize results using bar plots or summary tables

In [121…
```
import pandas as pd
import numpy as np

# Load embedded training examples (solved problems with embeddings)
df_embedded = pd.read_csv("/content/embedded_word_problems.csv")
df_embedded["embedding"] = df_embedded["embedding"].apply(eval).apply(np.arr

# Load test questions (no operation included)
df_test = pd.read_csv("/content/final_test_questions.csv")
```

In [122…
```
import openai
import os
import time

client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

def embed_question_openai(text, model="text-embedding-ada-002"):
    try:
        response = client.embeddings.create(input=[text], model=model)
        return np.array(response.data[0].embedding)
    except Exception as e:
        print("❌ Embedding error:", e)
        return None
```

In [123…
```
from sklearn.metrics.pairwise import cosine_similarity

def find_best_prompt(new_embedding):
    best_score = -1
    best_row = None

    for operation in df_embedded["operation"].unique():
        df_op = df_embedded[df_embedded["operation"] == operation].copy()
        df_op["similarity"] = df_op["embedding"].apply(
            lambda x: cosine_similarity([new_embedding], [x])[0][0]
        )
        top = df_op.sort_values("similarity", ascending=False).iloc[0]
        if top["similarity"] > best_score:
            best_score = top["similarity"]
            best_row = top

    return best_row
```

In [124…
```
def build_prompt(sim_q, sim_a, new_q):
    return f"""Here is a solved example:

Q: {sim_q}
A: {sim_a}

Now solve this similar problem:

Q: {new_q}
A:"""
```

In [125…
```python
from huggingface_hub import InferenceClient

llama_client = InferenceClient(
    provider="together",
    api_key=os.getenv("HF_API_KEY")
)

def query_llama4(prompt, max_tokens=256, temperature=0.7):
    try:
        start = time.time()
        result = llama_client.chat.completions.create(
            model="meta-llama/Llama-4-Scout-17B-16E-Instruct",
            messages=[{"role": "user", "content": prompt}],
            max_tokens=max_tokens,
            temperature=temperature
        )
        elapsed = time.time() - start
        return result.choices[0].message.content.strip(), elapsed
    except Exception as e:
        print("❌ API Error:", e)
        return None, None
```

In [126…
```python
def extract_numeric_answer(text):
    try:
        # Extract last integer from response
        nums = [int(s) for s in text.strip().split() if s.isdigit()]
        return nums[-1] if nums else None
    except:
        return None
```

In [127…
```python
results = []

for idx, row in df_test.iterrows():
    question = row["question"]
    true_answer = row["answer"]

    print(f"\n🧪 Processing Question {idx+1}/{len(df_test)}...")
    new_embedding = embed_question_openai(question)

    if new_embedding is None:
        print("⚠️ Skipping due to embedding error.")
        continue

    best_example = find_best_prompt(new_embedding)
    sim_q, sim_a = best_example["question"], best_example["answer"]
    operation_used = best_example["operation"]

    prompt = build_prompt(sim_q, sim_a, question)
    print("📤 Prompt Preview:\n", prompt[:300], "\n...")

    response, latency = query_llama4(prompt)

    predicted_numeric = extract_numeric_answer(response)
    correct = str(predicted_numeric) == str(true_answer)
    results.append({
        "question": question,
        "true_answer": true_answer,
        "predicted_answer": predicted_numeric,
        "correct": correct,
        "operation_used": operation_used,
        "prompt_sent": prompt,
        "model_response": response,
        "response_time": latency
    })
```

```
Streaming output truncated to the last 5000 lines.
📤 Prompt Preview:
 Here is a solved example:

Q: Ava has 48 candies and wants to divide them equally among 8 friends. How
many candies does each friend get?
A: 6

Now solve this similar problem:

Q: Evie bought 68 colorful candies for a party. After the party, 25 candies
were eaten by the kids. How many candies are le
...

🧪 Processing Question 606/1000...
📤 Prompt Preview:
 Here is a solved example:

Q: Mia has 12 boxes of books, each containing 4 books. How many books does M
ia have in total?
A: 48

Now solve this similar problem:

Q: Each evening, Luna reads 4 pages of a novel before bedtime. After reading
for 8 evenings, how many total pages has Luna read?
A:
...

🧪 Processing Question 607/1000...
📤 Prompt Preview:
 Here is a solved example:

Q: If Sara had 93 candies and gave away 74, how many candies does Sara have
left?
A: 19

Now solve this similar problem:

Q: Willow bought 74 colorful candies for a party. After the party, 66 candie
s were eaten by the kids. How many candies are left?
A:
...

🧪 Processing Question 608/1000...
📤 Prompt Preview:
 Here is a solved example:

Q: If Noah had 74 candies and gave away 18, how many candies does Noah have
left?
A: 56

Now solve this similar problem:

Q: Noah bought 65 colorful candies for a party. After the party, 47 candies
were eaten by the kids. How many candies are left?
```

```
Now solve this similar problem:

Q: Isla bought 97 colorful candies for a party. After the party, 93 candies
were eaten by the kids. How many candies are left?
A:
...

🧪 Processing Question 1000/1000...
📤 Prompt Preview:
 Here is a solved example:

Q: Mia has 2 boxes of books, each containing 7 books. How many books does Mi
a have in total?
A: 14

Now solve this similar problem:

Q: Each evening, Harvey reads 3 pages of a novel before bedtime. After readi
ng for 7 evenings, how many total pages has Harvey read?
A:
...
```

In [128…

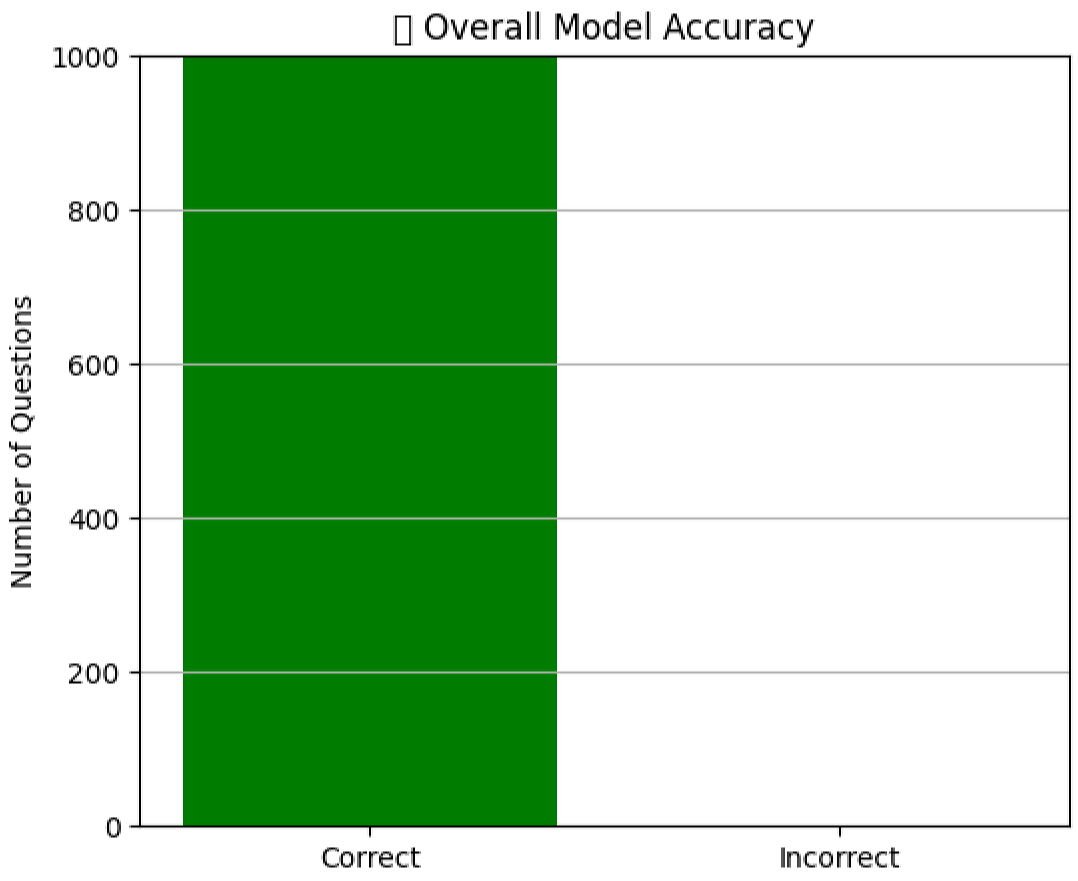
```
df_results = pd.DataFrame(results)
df_results.to_csv("/content/test_results.csv", index=False)
print("\n✅ All results saved to: /content/test_results.csv")
```

```
✅ All results saved to: /content/test_results.csv
```

In [129…

```
import pandas as pd
import matplotlib.pyplot as plt

# ✅ Load the test results
df = pd.read_csv("/content/test_results.csv")

# 🟦 Plot 1: Overall Accuracy
correct_count = df["correct"].sum()
incorrect_count = len(df) - correct_count

plt.figure(figsize=(6, 5))
plt.bar(["Correct", "Incorrect"], [correct_count, incorrect_count], color=["
plt.title("✅ Overall Model Accuracy")
plt.ylabel("Number of Questions")
plt.ylim(0, len(df))
plt.grid(axis='y')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: User
Warning: Glyph 9989 (\N{WHITE HEAVY CHECK MARK}) missing from font(s) DejaVu
Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



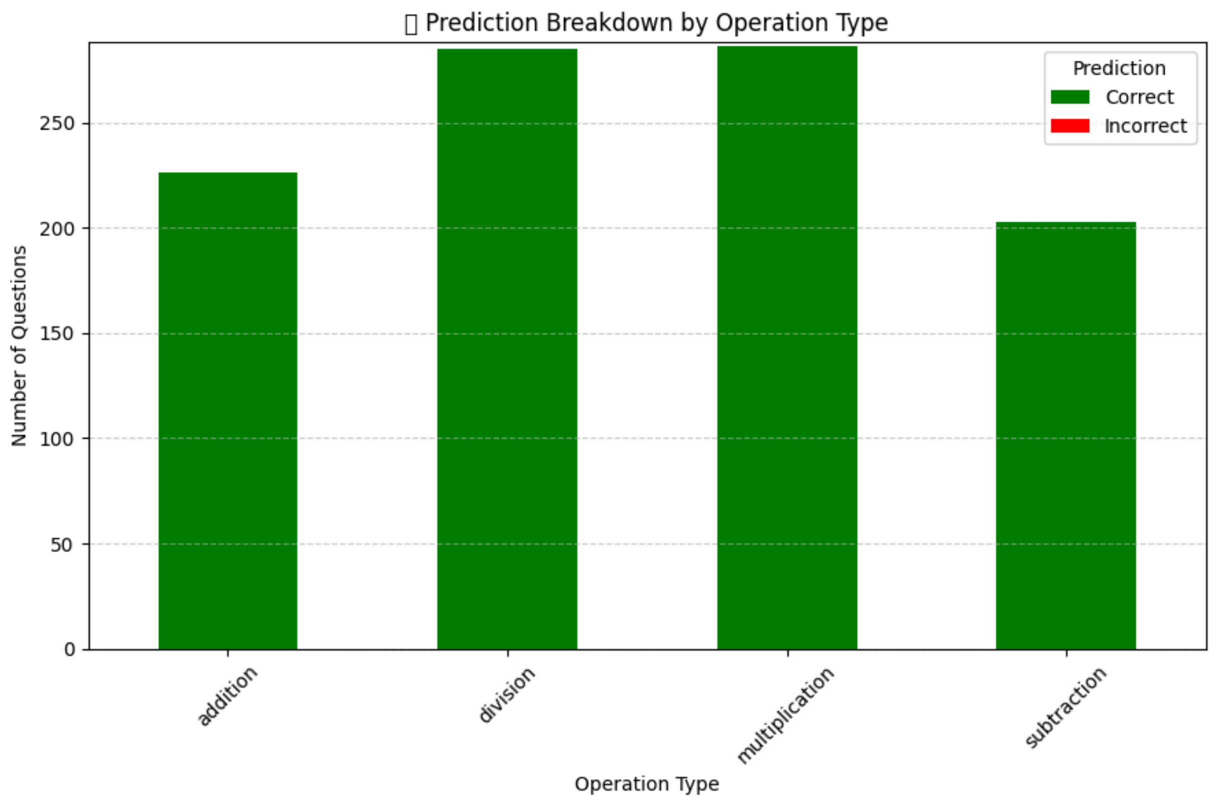
In [130…
```
import pandas as pd

# Load the test results
df = pd.read_csv("/content/test_results.csv")

# Total number of questions
total_questions = len(df)

# Number of questions per operation type
operation_counts = df["operation_used"].value_counts(dropna=False)

# Display results
print(f"📊 Total Questions: {total_questions}\n")
print("🧮 Questions per Operation Type:\n")
print(operation_counts)
```

```
📊 Total Questions: 1000

🧮 Questions per Operation Type:

operation_used
multiplication    286
division          285
addition          226
subtraction       203
Name: count, dtype: int64
```

In [131…
```
import pandas as pd
import matplotlib.pyplot as plt
```

```python
# Load your test results
df = pd.read_csv("/content/test_results.csv")

# Ensure correct column is int
df["correct"] = df["correct"].astype(int)

# Filter and group
df_filtered = df[df["operation_used"].notnull()]
summary = df_filtered.groupby("operation_used")["correct"].value_counts().ur

# Rename columns safely
column_mapping = {0: "Incorrect", 1: "Correct"}
summary.rename(columns=column_mapping, inplace=True)

# Ensure both columns exist
for col in ["Correct", "Incorrect"]:
    if col not in summary.columns:
        summary[col] = 0

# Reorder
summary = summary[["Correct", "Incorrect"]]

# Custom Y-axis limit: max number of questions per operation type
operation_counts = df_filtered["operation_used"].value_counts()
y_max = operation_counts.max() + 2  # add margin

# Plot
summary.plot(kind="bar", stacked=True, figsize=(9, 6), color=["green", "red"
plt.title("🔍 Prediction Breakdown by Operation Type")
plt.ylabel("Number of Questions")
plt.xlabel("Operation Type")
plt.xticks(rotation=45)
plt.legend(title="Prediction")
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.ylim(0, y_max)
plt.tight_layout()
plt.show()
```

```
<ipython-input-131-f22bbb00c633>:39: UserWarning: Glyph 128269 (\N{LEFT-POIN
TING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: User
Warning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font
(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



✅ Summary & Next Steps

- This notebook demonstrates embedding-based few-shot prompting.
- By retrieving similar examples, the model performs arithmetic reasoning better than random prompting.
- Future improvements could include:
  - Using multiple retrieved examples
  - Applying chain-of-thought reasoning
  - Training custom retrieval models

In [132…

```

```