Evaluation of Course Recommendation System

Your Name

August 1, 2024

1 Introduction

This document describes the methods and metrics used to evaluate a course recommendation system based on a Retrieval-Augmented Generation (RAG) pipeline. The evaluation focuses on comparing the *Generated Responses* produced by the system with the *Ideal Responses* for each prompt. Various metrics are calculated to assess the accuracy and relevance of the generated recommendations.

2 Data Description

The evaluation is performed using a dataset that contains prompts, ideal responses, and generated responses. The dataset is represented in a pandas DataFrame with the following columns:

- **Prompt ID**: A unique identifier for each prompt.
- **Prompt**: The query or question provided to the recommendation system.
- **Ideal Response**: The expected or ideal course recommendation for the given prompt.
- **Generated Response**: The course recommendation generated by the system.

3 Evaluation Metrics

The following metrics are used to evaluate the performance of the course recommendation system:

3.1 Exact Match

The **Exact Match** metric checks if the *Generated Response* is exactly the same as the *Ideal Response*, considering case insensitivity. It is calculated using the formula:

$$\text{Exact Match} = \begin{cases} 1, & \text{if the responses are exactly the same} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

3.2 BLEU Score

The **BLEU (Bilingual Evaluation Understudy) Score** is a metric that measures the precision of n-grams in the generated response compared to the ideal response. It evaluates how similar the generated response is to the ideal response by computing the geometric mean of the n-gram precisions and applying a brevity penalty for short responses.

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2}$$

where p_n is the precision of n-grams, w_n is the weight for each n-gram, and BP is the brevity penalty.

3.3 ROUGE Scores

The **ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Scores** measure the overlap between the generated response and the ideal response. We calculate the following ROUGE metrics:

- **ROUGE-1**: Overlap of unigrams (single words) between the generated and ideal responses.
- **ROUGE-2**: Overlap of bigrams (two consecutive words) between the generated and ideal responses.
- **ROUGE-L**: Longest Common Subsequence (LCS) between the generated and ideal responses.

The ROUGE score is calculated as follows:

$$\text{ROUGE-N} = \frac{\sum_{\text{n-gram} \in \text{Ideal}} \min(\text{Count}_{\text{n-gram}}(\text{Ideal}), \text{Count}_{\text{n-gram}}(\text{Generated}))}{\sum_{\text{n-gram} \in \text{Ideal}} \text{Count}_{\text{n-gram}}(\text{Ideal})} \tag{3}$$

3.4 Cosine Similarity

The **Cosine Similarity** metric calculates the cosine of the angle between two vectors representing the TF-IDF (Term Frequency-Inverse Document Frequency) representations of the ideal and generated responses. It measures the similarity between the two texts, where a value of 1 indicates identical responses and a value of 0 indicates no similarity.

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} \tag{4}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the TF-IDF vectors of the ideal and generated responses, respectively.

4 Methods to Improve RAG

Improving a Retrieval-Augmented Generation system involves optimizing both the retrieval and generation components. Below are some methods that can enhance the RAG system's performance:

4.1 Improving Retrieval

- **Enhanced Indexing Techniques**: Utilize advanced indexing algorithms to quickly locate relevant documents and contexts.
- **Contextual Embeddings**: Use transformer-based models to generate contextual embeddings that capture semantic meanings, improving retrieval accuracy.
- **Re-ranking Mechanisms**: Implement re-ranking strategies to prioritize the most relevant documents after initial retrieval.
- **Knowledge Graph Integration**: Leverage knowledge graphs to enrich context and improve retrieval with structured knowledge.

4.2 Improving Generation

- **Fine-Tuning Generative Models**: Fine-tune the generative model on domain-specific data to improve the quality of generated responses.
- **Prompt Engineering**: Carefully design prompts to guide the generation model in producing more accurate and context-aware responses.
- **Fact-Checking Mechanisms**: Integrate fact-checking tools to verify and enhance the factual correctness of generated responses.
- **Data Augmentation**: Use data augmentation techniques to increase the diversity and robustness of training datasets for the generative model.

5 Evaluation Results

The table below shows the evaluation results of the course recommendation system using the specified metrics.

Prompt ID	Prompt
1	Can you recommend a course that covers the basics of web development?
2	I'm interested in learning Python for application engineering. Can you suggest a course?
3	What course should I take if I want to understand data science engineering methods and tool
4	Do you have any courses that focus on business analysis and information engineering?
5	Can you suggest a course that teaches neural modeling methods and tools?
6	I'm a beginner with no programming experience. What courses should I start with?
7	Are there any courses that cover machine learning algorithms and their implementation?

Prompt ID	Prompt
8	Can you recommend a course on engineering high-performance coding for fintech?
9	Do you offer courses that provide an introduction to the fundamental techniques of web design and user
10	What course should I take if I want to learn about advanced big-data applications and indexing tec

6 Implementation

The evaluation is implemented using Python with the following key steps:

1. Import necessary libraries for text processing and evaluation, such as `pandas`, `nltk`, and `scikit-learn`.
2. Load the data into a pandas DataFrame, which includes prompts, ideal responses, and generated responses.
3. Define functions to calculate each evaluation metric.
4. Iterate over each row of the DataFrame, compute the metrics, and add them as new columns to the DataFrame.
5. Save the modified DataFrame to a CSV file for further analysis and reporting.

7 Conclusion

This document outlines the methods used to evaluate the performance of a course recommendation system. By comparing the generated responses with the ideal responses using various metrics, we can assess the system's accuracy and relevance. The results of this evaluation can inform further improvements to the system and help ensure high-quality course recommendations.

8 Code Appendix

Below is the Python code used for the evaluation:

Listing 1: Course Recommendation Evaluation

```
import pandas as pd
from nltk.translate.bleu_score import sentence_bleu
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from rouge_score import rouge_scorer

# Function to calculate Exact Match
def calculate_exact_match(ideal_response, generated_response):
```

```
    return 1 if ideal_response.strip().lower() == generated_response.strip().lower() else 0

# Function to calculate BLEU Score
def calculate_bleu_score(ideal_response, generated_response):
    reference = [ideal_response.split()]
    candidate = generated_response.split()
    return sentence_bleu(reference, candidate)

# Function to calculate ROUGE Scores
def calculate_rouge_scores(ideal_response, generated_response):
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
    scores = scorer.score(ideal_response, generated_response)
    return scores['rouge1'].fmeasure, scores['rouge2'].fmeasure, scores['rougeL'].fmeasure

# Function to calculate Cosine Similarity
def calculate_cosine_similarity(ideal_response, generated_response):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([ideal_response, generated_response])
    similarity = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
    return similarity[0][0]

# Calculate metrics for each row and add to the DataFrame
df["Exact Match"] = df.apply(lambda row: calculate_exact_match(row["Ideal Response"], row["G
df["BLEU Score"] = df.apply(lambda row: calculate_bleu_score(row["Ideal Response"], row["Gen
df["ROUGE-1"], df["ROUGE-2"], df["ROUGE-L"] = zip(*df.apply(lambda row: calculate_rouge_scor
df["Cosine Similarity"] = df.apply(lambda row: calculate_cosine_similarity(row["Ideal Respon

# Save the modified DataFrame to a new CSV file
df.to_csv("course_recommendation_evaluation_with_metrics.csv", index=False)

# Print the updated DataFrame to check the results
print(df)
```