20BBS0164
ARYAN KANDARI

Task Description: Building a Topic-based Post Recommendation System using Generative AI

Approach:
Core Components:
User Topic Profiling:
Objective: Create a user profile based on their interest, derived from their posts.
Method: Use Generative AI to analyze post content and identify up to 20 key topics (maximum 10 topics per user post).
Post Topic Profiling:
Objective: Assign topics to each post.
Method: Analyze posts to extract relevant topics, with the same topic limits as user profiles.
Recommendation Engine:
Objective: Match posts with the top 10 most relevant users based on their interests.
Approach: Use Generative AI to understand post and user profile topics and recommend posts to matching users.
Evaluation:
Metrics: Use Normalized Discounted Cumulative Gain (NDCG) and Jaccard similarity to evaluate the effectiveness of the recommendations.
Procedure: Compare the system's recommendations with the ground truth data in the testing set to measure accuracy and relevance.

1)To create a user profile based on their interests derived from their posts using Generative AI, you can follow these steps:

• Collect User Posts: Gather the posts made by the user that you want to profile. These posts can be in the form of text, such as social media posts, blog articles, or any other textual content.

• Preprocess the Text: Clean and preprocess the text of the user posts. This may involve removing special characters, punctuation, stopwords, and performing tokenization and stemming.

• Topic Modeling: Apply a topic modeling technique, such as Latent Dirichlet Allocation (LDA) or Non-Negative Matrix Factorization (NMF), to the preprocessed text. These techniques can automatically identify key topics present in the user posts.

• Train the Generative AI Model: Use the preprocessed text as input to train a Generative AI model, such as a language model or a sequence-to-sequence model. This model should be capable of generating text based on the given input.

• Generate Topics: Utilize the trained Generative AI model to generate up to 10 topics for each user post. The model should be able to generate relevant and coherent topics based on the content of the posts.

• Topic Aggregation: Aggregate the generated topics from all the user posts to create a comprehensive list of topics. Remove any duplicate or similar topics to ensure a concise and diverse representation of the user's interests.

• Profile Creation: Create a user profile based on the aggregated topics. This profile can include the top 20 topics that represent the user's interests. You can also assign weights or scores to each topic based on their frequency or importance in the user's posts.

2) To match posts with the top 10 most relevant users based on their interests using Generative AI, you can follow these steps:

• User Profile Creation: Create user profiles for each user by assigning topics to their posts, as described in the previous steps. Each user profile should contain the top 20 topics that represent their interests.

• Post Topic Extraction: Analyze the content of each post to extract relevant topics, as described in the previous steps. Assign the generated topics to each post.

• Generative AI Model Training: Train a Generative AI model, such as a language model or a sequence-to-sequence model, using the user profiles and the assigned topics from the posts. The model should be capable of understanding the topics and generating text based on the given input.

• Post-User Matching: For each post, utilize the trained Generative AI model to match the post with the top 10 most relevant users based on their interests. This can be done by comparing the topics of the post with the topics in the user profiles and calculating a relevance score or similarity measure.

• Recommendation Output: Store the matched post-user pairs in a structured format, such as a database or a JSON file. Each post should be associated with the top 10 most relevant users who have similar interests based on their user profiles.

3) To evaluate the effectiveness of the recommendations using Normalized Discounted Cumulative Gain (NDCG) and Jaccard similarity, you can follow these steps:

• Testing Set Preparation: Prepare a testing set that contains ground truth data, which includes the actual posts that users found relevant or engaged with. Each post in the testing set should be associated with the users who found it relevant.

• Recommendation Generation: Generate recommendations for each user based on their interests and the posts they have not yet engaged with. Use the approach described earlier to match posts with the top 10 most relevant users.

• NDCG Calculation: Calculate the Normalized Discounted Cumulative Gain (NDCG) for the recommendations. NDCG measures the ranking quality of the recommended posts by comparing them to the ground truth data. It takes into account both the relevance of the recommended posts and their ranking position. Higher NDCG scores indicate more accurate and relevant recommendations.

• Jaccard Similarity Calculation: Calculate the Jaccard similarity between the recommended posts and the ground truth data. Jaccard similarity measures the overlap between the recommended posts and the posts that users found relevant. It provides a measure of how well the recommendations match the user's interests. Higher Jaccard similarity scores indicate more relevant recommendations.

• Evaluation Output: Store the NDCG scores and Jaccard similarity scores for the recommendations. You can aggregate these scores across all users in the testing set to get an overall evaluation of the system's effectiveness.

4) To integrate Wikipedia topic verification into your project, you can use the Langchain library, specifically the WikipediaRetriever class. Here are the steps to follow:

• Install the Langchain library by running the following command:
pip install langchain

• Import the WikipediaRetriever class into your project:
from langchain.retrievers import WikipediaRetriever

• Create an instance of the WikipediaRetriever class:
retriever = WikipediaRetriever()

• Use the get_relevant_documents() method to retrieve relevant Wikipedia articles based on a query:
query = "your topic query"
docs = retriever.get_relevant_documents(query=query)

• Access the metadata and page content of the retrieved documents:

```
for doc in docs:
    metadata = doc.metadata
    page_content = doc.page_content
    # Use the metadata and page content for topic verification
```

GITHUB LINK:
https://github.com/aryankandari/NLP2-farmwise-task