PROJECT PRESENTATION

NATURAL LANGUAGE PROCESSING

TEAM - 23

“Semantic Search Engine for Movies”

Team Members

- Chaitanya - se22uari047
- Shivaranjani - se22uari158
- Siddhartha - se22uari063
- Sriram - se22uari070

Team Contribution:

- shivaranjani: Data preprocessing , implemented text cleaning.
- chaitanya: Integrated Sentence-BERT and FAISS for semantic search.
- sriram: Hyperparameter tuning with Optuna; optimized model performance.
- siddhartha: Evaluated metrics (Precision, Recall, nDCG, MRR); visualized results.
- Each member contributed equally to the project’s success.

Abstract

This project develops an advanced movie recommendation system that leverages the power of natural language processing (NLP) and deep learning techniques to recommend films based on user queries. Using the IMDb Top 1000 movies dataset, we employ the Sentence-BERT model to generate text embeddings, facilitating semantic search. By preprocessing the movie descriptions, genres, and cast information into a unified text format, the system can understand and match user queries to relevant movies. The system is optimized for performance with hyperparameter tuning using Optuna, ensuring that the most accurate movie recommendations are retrieved. Evaluation metrics such as Precision@k, Recall@k, Mean Reciprocal Rank (MRR), and nDCG@k are used to assess the system's effectiveness, providing a comprehensive view of its recommendation quality.

Introduction

Movie recommendation systems enhance user experiences by suggesting films based on personal preferences. This project builds a recommendation system using the IMDb Top 1000 dataset, which includes details like movie titles, genres, and cast. By preprocessing and combining text data into a single string, we convert it into embeddings using Sentence-BERT, enabling semantic search for relevant movie recommendations. FAISS is used for fast indexing and retrieval, while Optuna optimizes the model's hyperparameters. Evaluation metrics such as Precision@k and nDCG@k assess the system's effectiveness, aiming to provide accurate and relevant suggestions to users.

Prior Related Work

Movie recommendation systems have evolved from basic collaborative filtering to advanced methods using deep learning and NLP. Early systems like Netflix's relied on collaborative filtering, while recent approaches use Sentence-BERT to improve semantic understanding of movie descriptions. FAISS is often used for fast similarity searches. These advancements address challenges like the cold start problem, and our approach builds on these, using Sentence-BERT and FAISS to enhance recommendation accuracy and speed.

Pipeline

- Data Collection
- Data Preprocessing
- Model selection
- Embedding Generation
- Similarity Search
- Hyperparameter Tuning
- Testing
- Recommendation

Dataset:

IMDb dataset is used to gather movie details, including genres, ratings, and cast.

link for the dataset
https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows

Statitics of dataset

Basic Information:

- **Total Entries**: 1000 rows.
- **Total Columns**: 16 columns, including `Poster_Link`, `Series_Title`, `Released_Year`, `Certificate`, `Runtime`, `Genre`, `IMDB_Rating`, `Overview`, `Meta_score`, `Director`, `Star1`, `Star2`, `Star3`, `Star4`, `No_of_Votes`, and `Gross`.

Column Statistics:

- **Series_Title**: 999 unique movie titles with one duplicate.
- **Released_Year**: Spread across 100 unique years, with 2014 being the most frequent year.
- **Certificate**: 16 unique certificates, with "U" as the most common.
- **Runtime**: Ranges across 140 unique values, with "100 min" as the most frequent.
- **Genre**: Highly diverse with 202 unique genre combinations, with "Drama" as the most common.
- **IMDB_Rating**: Mean rating is 7.95, with values ranging from 7.6 to 9.3.
- **Meta_score**: Available for 843 movies, with a mean of 78 and a range of 28 to 100.
- **Director**: 548 unique directors, with Alfred Hitchcock appearing most frequently.
- **Stars**: Tom Hanks, Emma Watson, Rupert Grint, and Michael Caine are some of the most frequent lead actors.
- **No_of_Votes**: Average of 273,693 votes, ranging up to 2,343,110.
- **Gross**: Contains 831 non-null values, with the most frequent entry being 4,360,000

DATA PRE-PROCESSING

In this phase, we focus on transforming the raw data into a clean format suitable for model training and embedding generation.

- Combining Relevant Information
  We created a new column that combines multiple important movie details such as the title, overview, genre, director, and cast. This consolidated text provides a rich description of each movie, which will be used to generate embeddings.

- Text Cleaning
  A preprocessing function was applied to the combined text:
  - Lowercasing: All text was converted to lowercase to ensure consistency.
  - Whitespace Removal: Extra spaces and unnecessary white spaces were removed to standardize the text.

Embedding Generation

We now have a processed_text column that contains clean, lowercase, and whitespace-free text for each movie, ready to be used for embedding generation and downstream tasks.

To convert our textual data into a machine-readable format for NLP tasks, we used Sentence-BERT for generating embeddings.

- **Model Selection**
- We used the pre-trained Sentence-BERT model, specifically the 'all-MiniLM-L6-v2' variant, which is lightweight and effective for generating sentence-level embeddings.
- **Generating Embeddings**
- For each movie's processed text (which includes title,
- genre, overview, etc.), we applied the model to generate dense vector embeddings. These vectors represent the semantic meaning of the movie descriptions.
- **Storage**
- The generated embeddings were stored in the 'embeddings' column of the dataset, making it ready for similarity comparison or clustering tasks.

Semantic Search with FAISS

FAISS Indexing

We convert movie embeddings into a NumPy array and add them to a FAISS index using L2 distance for efficient semantic search.

Search Function

The search_movie() function:

- Converts the query into an embedding.
- Finds the top K similar movies using FAISS.
- Returns movie details like title, overview, genre, and IMDB rating.

Why BERT?(research part)

BERT (Bidirectional Encoder Representations from Transformers) is ideal for your Semantic Search Engine for Movies because:

- Contextual Understanding
- Sentence-Level Representations
- Pretrained Strength
- Versatility
- Encoder-Only Model Architecture

Contextual Understanding:

- Unlike earlier models, BERT considers the context of a word by analyzing both the words before and after it in a sentence. This bidirectional approach is crucial for semantic search, as it captures the nuanced meanings of queries and movie descriptions.

Sentence-Level Representations:

- Sentence-BERT (a variation of BERT) is specifically designed to produce meaningful embeddings for entire sentences. This is perfect for comparing user queries with movie metadata or descriptions in a semantic way.

Pretrained Strength:

- BERT is pretrained on massive datasets, providing a strong foundation for fine-tuning on domain-specific data like movie titles, genres, or plot summaries.

Versatility:

- BERT excels in various NLP tasks like text classification, similarity detection, and question answering—all relevant for building a robust search engine.

BERT's Encoder-Only Model Architecture

- Purpose: Focuses on understanding and encoding the input text (queries or movie descriptions) into rich, contextual embeddings.
- How It Works:
- The encoder takes a sequence of tokens (words or subwords) as input and applies self-attention to learn the relationships between all tokens in the sequence.
- Example: In "A movie about friendship and betrayal", BERT understands how "friendship" and "betrayal" relate within the sentence.

2. Transformer Encoder Layers

BERT uses multiple layers of the encoder, each consisting of:

- Self-Attention Mechanism:
  - This computes how each word relates to every other word in the sentence, capturing contextual meaning.
  - Example: In "a funny action movie", it identifies that "funny" modifies "movie", while "action" provides genre context.
- Feed-Forward Networks:
  - These process the self-attention outputs to refine the understanding of each word's role.

3. Bidirectional Context Understanding

- BERT’s encoder processes input text in both directions (left-to-right and right-to-left) simultaneously.
  - Advantage: It captures how the context before and after a word influences its meaning.
  - Example: "dark comedy" in "a dark comedy with humor" has a different meaning than in "dark comedy about crime."

Why Encoder-Only for Semantic Search?

- Efficient Representation: The encoder generates embeddings that summarize the meaning of queries or movie descriptions into fixed-length vectors.
- Comparison:
- User query embeddings (e.g., "adventure movie with a hero") can be directly compared with movie metadata embeddings to find the most relevant results.

How BERT is Different from Word2Vec and Similar Models?

Feature	Word2Vec/GloVe	BERT
Embedding Type	Static: Each word has a single, fixed vector.	Contextual: Word embeddings change based on the surrounding words.
Contextual Understanding	Ignores context; treats words independently.	Considers the full sentence context (bidirectional).
Out-of-Vocabulary Words	Fails to represent unseen words.	Uses subword tokenization (WordPiece) to handle rare words.
Sentence-Level Meaning	Not designed for sentence-level embeddings.	Sentence-BERT creates embeddings for entire sentences or paragraphs.
Applications	Basic similarity tasks or clustering.	Advanced NLP tasks like semantic search, question answering, and more.

Difference From Encoder-Decoder Models

Aspect	BERT (Encoder-Only)	Encoder-Decoder Models (e.g., T5)
Focus	Text understanding (e.g., semantic search, classification).	Text-to-text tasks (e.g., translation, summarization).
Output	Contextual embeddings for the input.	Generates new text as output.
Use Case	Perfect for similarity tasks like semantic search.	Better suited for tasks requiring text generation.

Hyperparameter Tuning with Optuna

We use Optuna to optimize hyperparameters for semantic search. Key hyperparameters include:

- Model: all-MiniLM-L6-v2 or all-mpnet-base-v2
- Top-k: Number of similar movies (3 to 10)
- Similarity Metric: L2 (Euclidean) or Inner Product

Results!

The best hyperparameters obtained from the tuning process are:

- Model: all-MiniLM-L6-v2
- Top-k: 10 (number of similar movies to retrieve)
- Similarity Metric: Inner Product (IP)

EVALUATION

- Precision@k: Measures the proportion of relevant items in the top-k results.
- Recall@k: Measures how many relevant items are retrieved in the top-k results.
- MRR (Mean Reciprocal Rank): Averages the reciprocal rank of the first relevant item.
- nDCG@k: Evaluates ranking quality by considering the position of relevant items.

Evaluation Process:

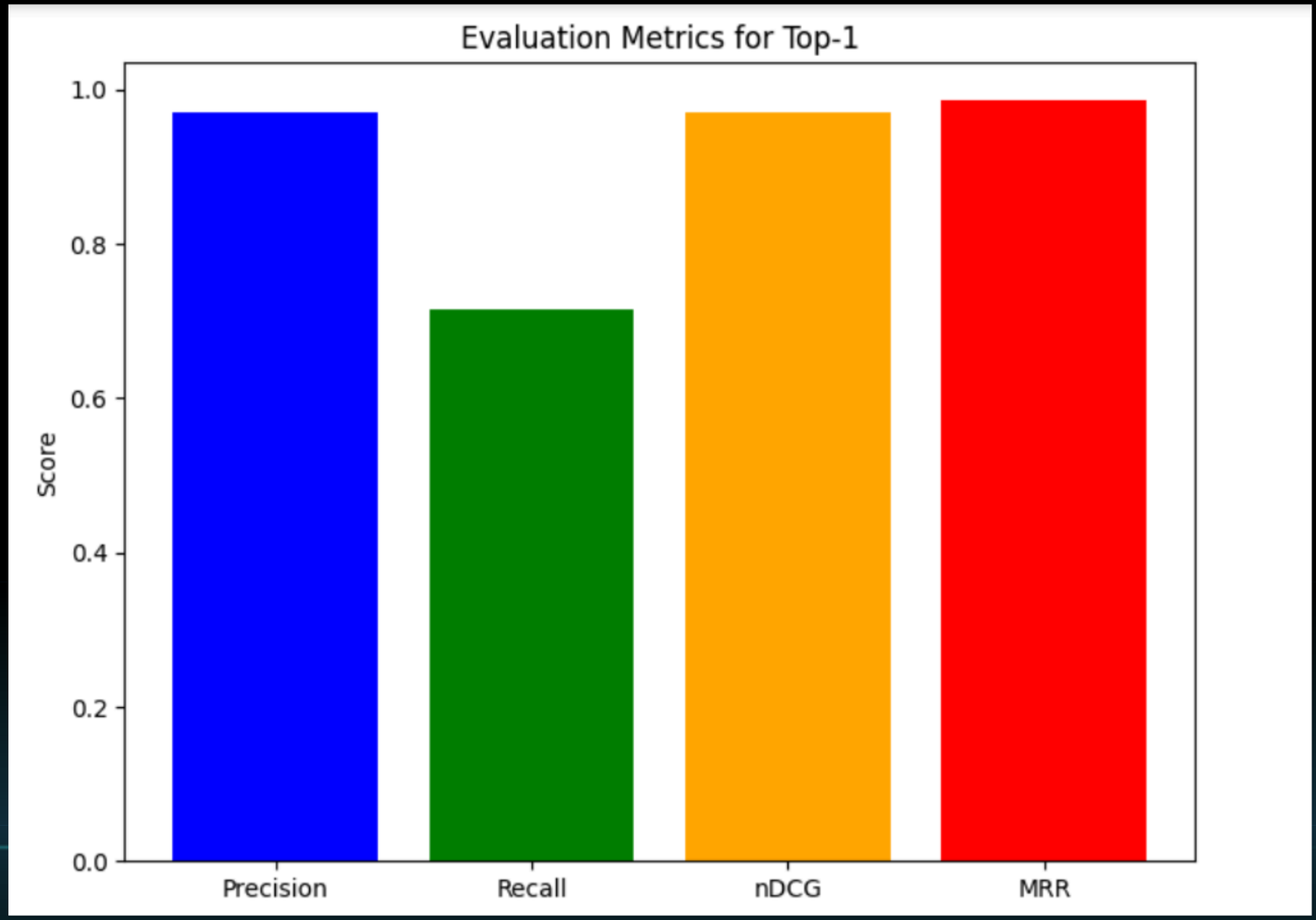
- A list of test queries (test_queries) is processed, and the top-k results are retrieved for each.
- For each query, the evaluation metrics (Precision@k, Recall@k, nDCG@k, and MRR) are computed based on the comparison with the ground truth relevant items (ground_truth_relevant).
- These metrics are then averaged and printed for Top-1 results (or other specified k values), with the final scores stored for plotting and further analysis.

Analysis & Conclusion:.

Based on the evaluation metrics for Top-1, here are the results:

- Precision@1: 0.9714 – This indicates that 97.14% of the retrieved results at the top position are relevant.
- Recall@1: 0.7143 – This means 71.43% of the relevant items were retrieved at the top position.
- nDCG@1: 0.9714 – The normalized Discounted Cumulative Gain at Top-1 is high, showing that the relevant results are ranked highly.
- MRR: 0.9857 – The Mean Reciprocal Rank is very high, suggesting that the first relevant result is retrieved almost always at the top.

GRAPH:


Evaluation Metrics for Top-1
Score
1.0
0.8
0.6
0.4
0.2
0.0
Precision
Recall
nDCG
MRR

Reference and Acknowledgment

The content and methodologies in this work were guided by the course schedule and materials provided by AI Course Schedule - Fall 2024.
implementation of faiss : link

Thank You