MM.DD.YY

PodMatch : Intelligent Search Engine for Podcast

Apoorva Reddy Bagepalli
Master's Student
Major : Data Science

Lorem Ipsum

1. Defining the Problem
2. Making Observations
3. Tech Stacks Used
4. Experiment Results
5. Live Demo Link

1. **Defining the Problem**
2. Making Observations
3. Tech Stacks Used
4. Experiment Results
5. Live Demo Link

Podcasts have become a popular medium for sharing knowledge and stories, offering a vast amount of valuable content. However, their unstructured and long-form nature creates significant challenges in content discoverability. Listeners often remember specific keywords or phrases from a podcast but struggle to identify the corresponding episode or title. This lack of efficient search tools hinders users from accessing relevant information quickly and prevents podcast creators from fully engaging their audience.

1. Defining the Problem
2. **Making Observations**
3. Tech Stacks Used
4. Experiment Results
5. Live Demo Link

This project presents a podcast search engine that combines classical and modern NLP retrieval models to handle both keyword-based and semantic queries. It supports several retrieval strategies, giving users the flexibility to search using the most suitable approach for their needs: Retrieval Models Implemented:

BM25: Classic bag-of-words based scoring using term frequency and document frequency.

TF-IDF: Vector space model scoring documents based on term importance.

BM25 + TF-IDF (Hybrid Lexical): Weighted ensemble that balances precision from BM25 and coverage from TF-IDF.

BERT + BM25 (Hybrid Semantic): Combines contextual embeddings from Sentence-BERT with BM25 to balance semantics and exact term match.

BERT + Self-Attention: A custom-built transformer variant that adds a multi-head attention layer on top of BERT token embeddings to improve representation.

RoBERTa: Dense retrieval using the roberta-base transformer for contextual embeddings.

Encoder-Based Retrieve & Re-Rank: A two-stage pipeline:

Stage 1: Fast retrieval using a Bi-Encoder.

Stage 2: Reranking candidates using a Cross-Encoder for precise relevance scoring. This ensemble of models ensures that the search engine supports both exact-match and semantic-match scenarios, catering to different types of users and queries. Whether the user knows the exact phrase or just remembers the theme, the engine can find the best-matching content.

1. Defining the Problem
2. Making Observations
3. **Tech Stacks Used**
4. Experiment Results
5. Live Demo Link

Languages: Python

Libraries: SentenceTransformers, transformers, faiss for vector search scikit-learn for TF-IDF and cosine similarity

rank_bm25 for probabilistic BM25

nltk, spaCy, contractions for preprocessing

Streamlit for the web-based interface

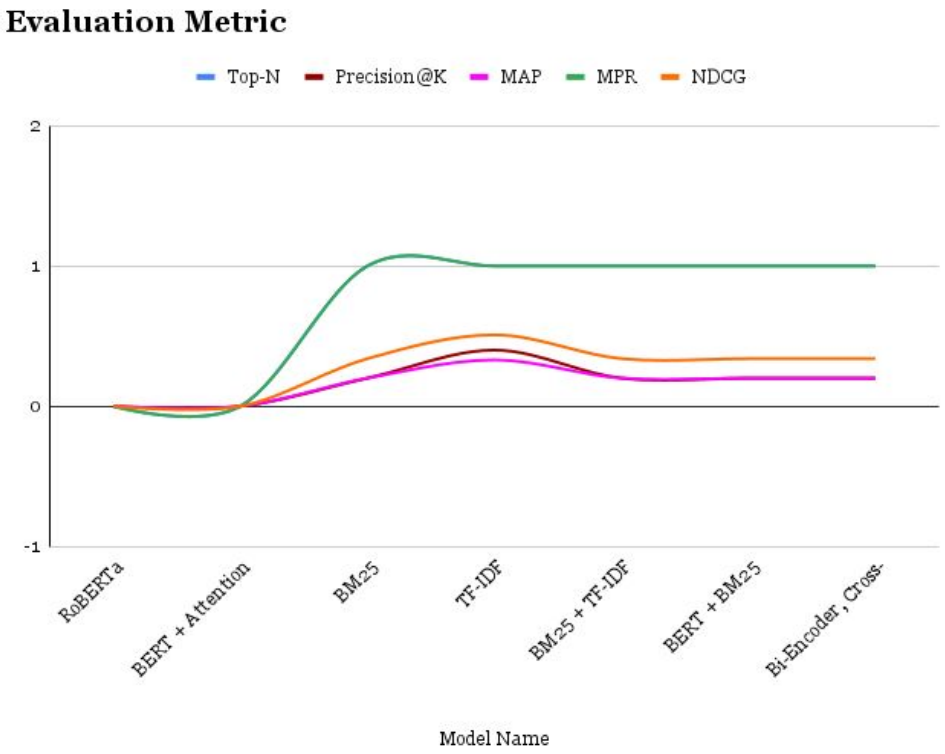
Models: all-MiniLM-L6-v2, roberta-base, custom BERT + Self-Attention

cross-encoder/ms-marco-MiniLM-L-6-v2 for reranking

Precomputed FAISS indexes and NumPy embeddings for fast retrieval

1. Defining the Problem
2. Making Observations
3. Tech Stacks Used
4. **Experiment Results**
5. Live Demo Link

- TF-IDF delivered the strongest overall performance, leading in Precision@K, MAP, and NDCG—highlighting its effectiveness for this domain.
- BM25 and hybrid models (BM25 + TF-IDF, BERT + BM25) showed stable but unimproved results, suggesting the need for better weighting or integration strategies.
- RoBERTa and the Self-Attention Transformer didn't yield strong results, possibly due to limited training data or lack of domain-specific fine-tuning—despite their strong semantic awareness.
- Bi-Encoder + Cross-Encoder architecture matched sparse retrieval models, indicating a solid foundation but room for improvement in capturing fine-grained relevance.
- Semantic models show promise but may need larger datasets, more domain-aligned queries, or enhanced training to unlock their full potential.



1. Defining the Problem
2. Making Observations
3. Tech Stacks Used
4. Experiment Results
5. **Live Demo Link**