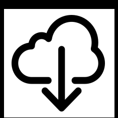
DEEP-POD
An enhanced podcasting experience

Mina Sonbol

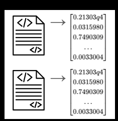
- Try it here!
-

Hot it works

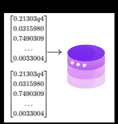


Step 1 - The podcast episode is downloaded.

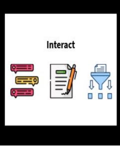
Step 2 - The episode is transcribed either using the incredibly-fast-whisperer hosted by Replicate, or using fast-whisperer locally (works best with a GPU).



Step 3 - The text embeddings are extracted using TF-IDF, or vector embeddings.



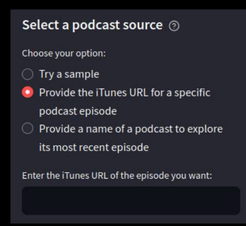
Step 4 - The text embeddings are stored into a vector database.



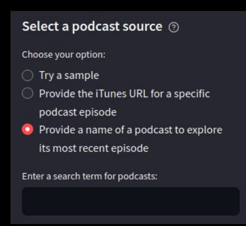
Step 5 - Interact with the podcast! A RAG pipeline supports this functionality.

Usage

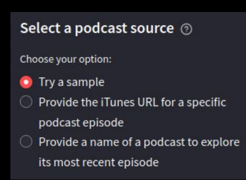
Users can provide a URL to a specific episode.



Insert the name of a podcast to get its most recent episode.

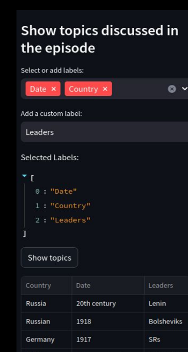


Alternatively, users can try out a sample episode



Interact

Deep-pod is an app that allows users to interact with their favorite podcasts in new ways through chat, summarization and topic extraction.



✍️ Transcription

The tool uses different flavors of OpenAI's open-sourced Whisper model for transcription. Users can either use the incredibly-fast-whisperer model hosted by Replicate (will require an API key), or they can run the fast whisper model locally, however, that mode is best suited for systems with GPUs.

Method		Replicate	
Speed	🏃🏃🏃	🏃🏃	🏃
RTF†	0.02	0.11	0.54
⏱️‡	1.2	6.6	32.4

† Real-time factor, calculated using this formula:

$$RTF = \frac{Transcription\ time}{Audio\ length}$$

‡ Time to transcribe a 1-hour episode in minutes

🧬 Embeddings

The tool provides two vectorization options, TF-IDF, and vector embeddings.

TF-IDF relies on word counts to determine word relevance, while embeddings are language models - usually encoder-decoder models - that are trained with the specific goal of creating text vectors that capture the semantic meaning of the text.

Users can use one of two embedding models: T5 (open source) and OpenAI's embeddings 3 (requires an API key).

⏱️ to embed 1000 words

T5 → 5.63 seconds
OpenAI → 41.06 seconds

Dimensions

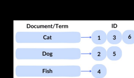
T5 → 768
OpenAI* → 3072

* The embeddings 3 model was trained for flexibility, that is, utilizing less dimensions should not impact performance. However, in practice that was not the case, using less embeddings did have a negative impact on semantic search results.

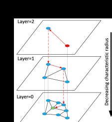
📖 Indexing

Once the text embeddings are created they are stored in a vector database that will serve as the data source from which the tool will generate responses to user queries. The tool provides two vector database options, Elasticsearch and ChromaDB.

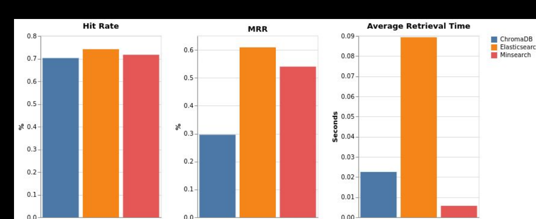
Elasticsearch indexes are designed to scale horizontally, and each index is broken down into smaller chunks called shards, that are distributed across the nodes for better performance and scalability. It uses inverted indexes for efficient term matching across documents.



ChromaDB indexes on the other hand are designed to scale vertically. It uses Hierarchical Navigable Small World graphs to navigate high-dimensional vector spaces swiftly, providing lightning fast vector searches.



Minsearch uses TF-IDF vectors and stores them in a pandas DF. It is not a scalable solution, implemented for demonstration purposes only.



The above visualization compares the 3 indexes across 3 metrics, Hit-Rate, MRR, and Average retrieval time.

Hit-Rate (aka Recall) → $\frac{Number\ of\ Hits}{Total\ Number\ of\ Requests}$

MRR (Mean Reciprocal Rank) → $\frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$

Retrieval Augmented Generation (RAG)

The RAG pipeline has 3 steps, searching the vector database for documents relevant to the user's query, building the prompt, and finally, text generation.

Search

1- 💬 User enters query
2- 🧬 Query is encoded
3- 🔎 The index search is conducted
4- 🧮 Cosine similarity is used to determine relevance.

Prompt

The prompt contains 3 parts:
1- 📋 The instructions
2- ❓ The query
3- 📚 The context

Respond

The prompt is then passed to an LLM to generate an answer. There are two LLM options, GPT-4o and FLAN5

Evaluation

The RAG pipeline is evaluated using an LLM-as-a-judge. A sample of 200 questions is passed to each LLM and the evaluator determines whether the answer is: relevant, partly relevant, or not relevant

GPT-4o

Relevant	54.5%
Partly-relevant	36.0%
Not-relevant	9.50%

FLAN-5

Relevant	0.00%
Partly-relevant	61.0%
Not-relevant	39.0%