Knowledge Extraction from Podcasts

Dataset Overview

Lex Fridman Podcast playlist on Youtube

- 75 episodes with well-defined timestamps (as on 11/22/2020), each around 2-4 hours.
- Dynamic data, with 2 to 3 new podcasts added every week.

Unstructured raw data

- Audio files

Structured metadata

- Text files

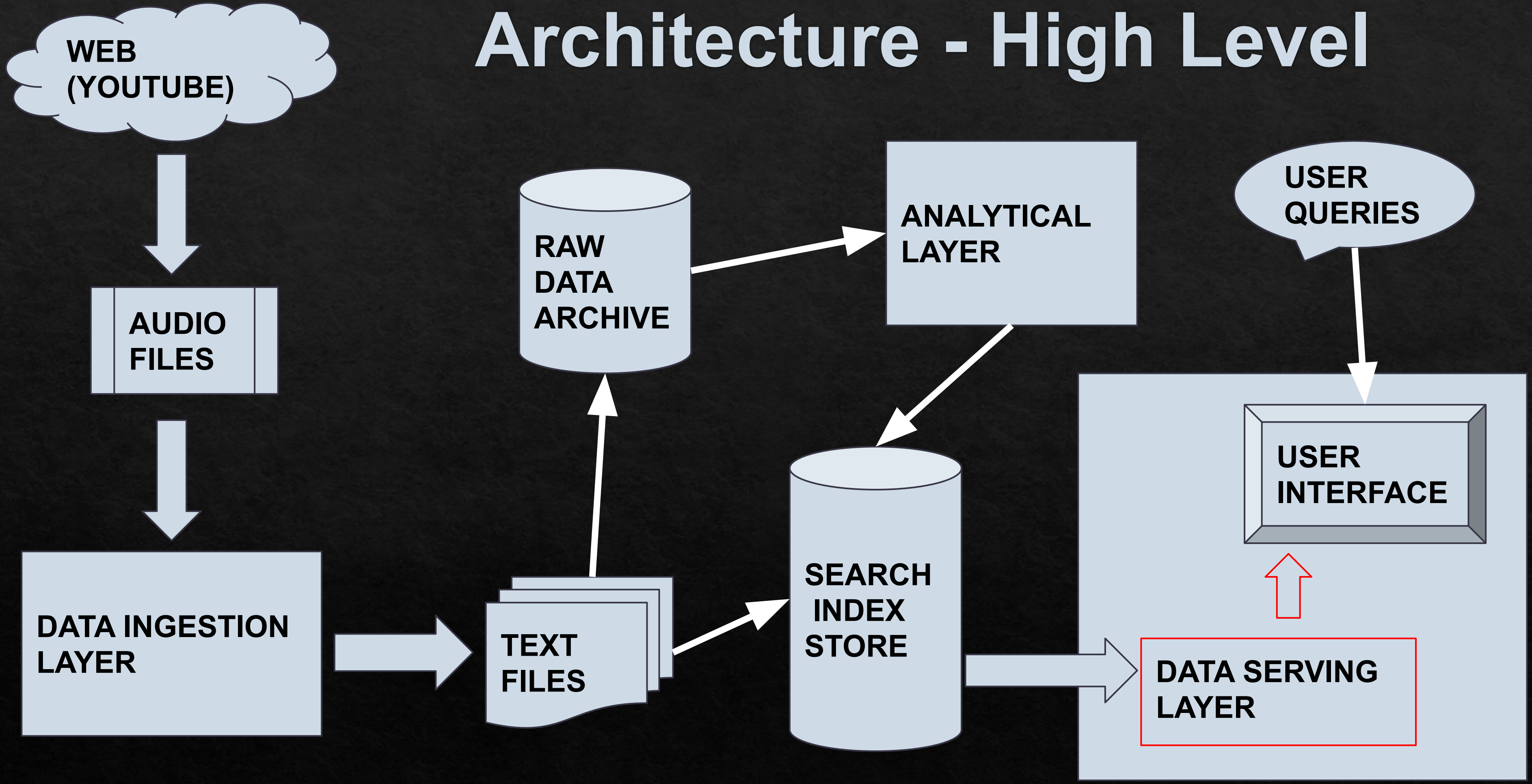
Why is the problem / dataset interesting?

- As per the internet, 850,000 active podcasts and 30 million episodes on the web, impossible to listen to them all for a user

- As opposed to mainstream media, contain invaluable information, expert opinions and diverse perspectives, ranging from science, society, politics, to life and beyond

- By efficiently organizing and processing this huge data, we can create an easily explorable repository of knowledge for end-users

Architecture - High Level



Architecture - End to End

DATA INGESTION LAYER

- EXTRACT
  - *Data Downloader* : downloads Youtube videos as .wav format audio files and metadata for videos as text files

- TRANSFORM
  - *Audio segmentation by subtopics* : uses timestamps in metadata to split full podcast audio into multiple shorter segments, each covering a subtopic of discussion

  - *Speaker Diarization* : identifies timestamp partitions for each subtopic segment audio stream based on speaker identity

Architecture - End to End

DATA INGESTION LAYER

- TRANSFORM
  - *Audio segmentation by speakers* : uses speaker diarization information to split each subtopic segment audio into multiple shorter speaker segments

- LOAD
  - *Raw Data Load* : transcribes audio to text for each speaker segment and writes each segment as one textual record to the raw data archive

  - *Index Segments* : summarizes each speaker segment text (transcribed utterances) and writes each summarized speaker segment as one record to the search index store

Architecture - End to End

DATA INGESTION LAYER

- LOAD
  - *Index Metadata and Speakers* : writes metadata to separate indices in the search index store as one record for each podcast as well as for each speaker

ANALYTICAL LAYER

- *Setup Knowledge Graph* : performs named entity recognition over each speaker segment in raw data store and defines speaker to entity relationships to realize a knowledge graph

- *Index Knowledge Graph* : writes all nodes and edges of the knowledge graph as records in two separate indices in the search index store

Architecture - End to End

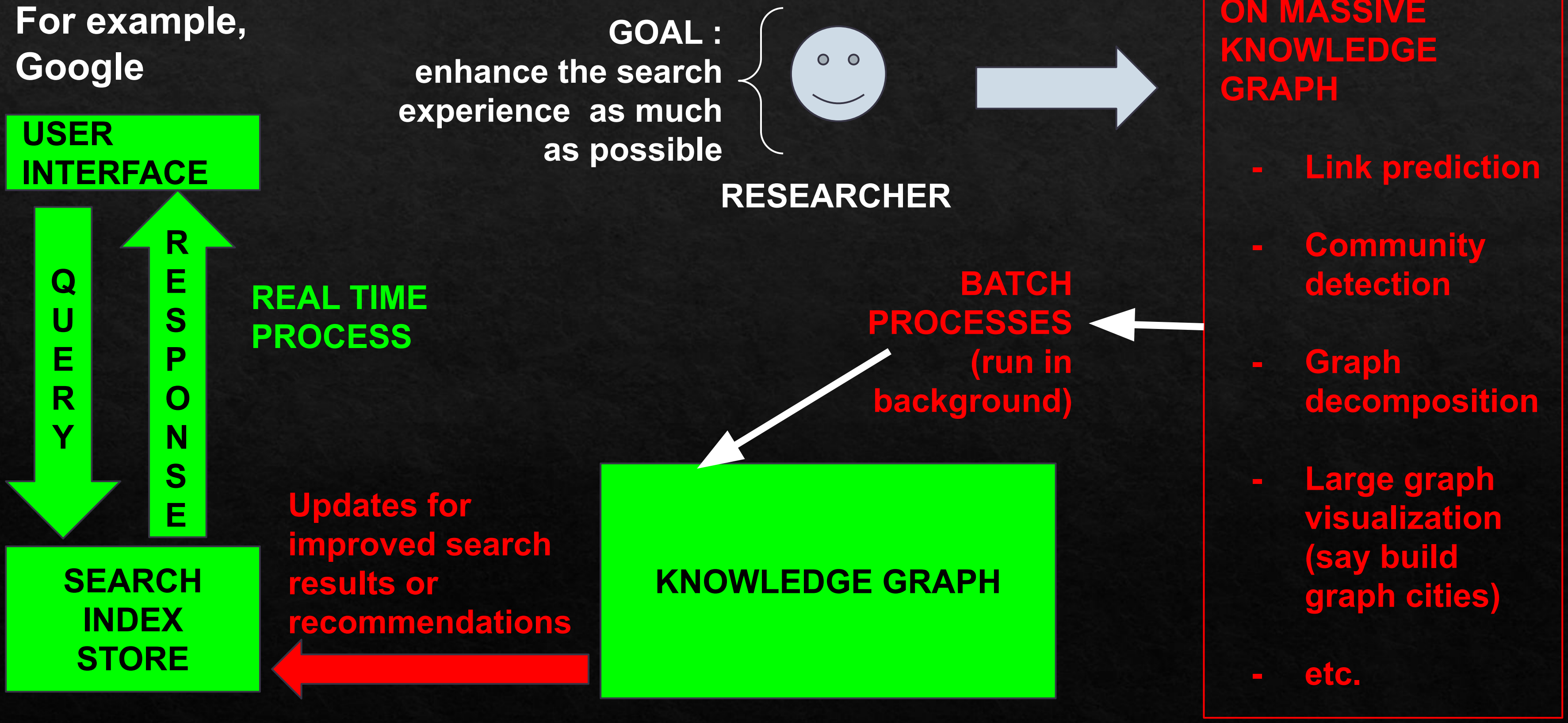
DATA SERVING LAYER

- User interface to the end-user for searching the knowledge repository built over podcasts data
- REST endpoints to query the search index store for retrieving processed data
- Knowledge graph visualization

Current scope of project

YES

NO



For example, Google

USER INTERFACE

QUERY

RESPONSE

REAL TIME PROCESS

SEARCH INDEX STORE

Updates for improved search results or recommendations

KNOWLEDGE GRAPH

GOAL : enhance the search experience as much as possible

RESEARCHER

BATCH PROCESSES (run in background)

EXPERIMENTATION ON MASSIVE KNOWLEDGE GRAPH

- Link prediction
- Community detection
- Graph decomposition
- Large graph visualization (say build graph cities)
- etc.

Tools & Libraries

- Programming Languages
  - Backend : Python and Java
  - Frontend : HTML and JavaScript

- Data Lake
  - Raw data archive : MongoDB
  - Search index store : Elasticsearch

- Distributed computing / Data wrangling
  - PySpark and Pandas

- Data Download (Data Ingestion Layer)
  - pytube + ffmpeg

- Audio Segmentation (Data Ingestion Layer)
  - pydub

Tools & Libraries

- Speaker Diarization (Data Ingestion Layer)
  - CMU Sphinx (Java)

- Audio to text transcription (Data Ingestion Layer)
  - Mozilla deepspeech

- Text summarization (Data Ingestion Layer)
  - bert-extractive-summarizer

- Named Entity Recognition (Analytical Layer)
  - spacy

- Web application (Data Serving Layer)
  - flask

- Graph visualization (Data Serving Layer)
  - D3.js (force directed graph)

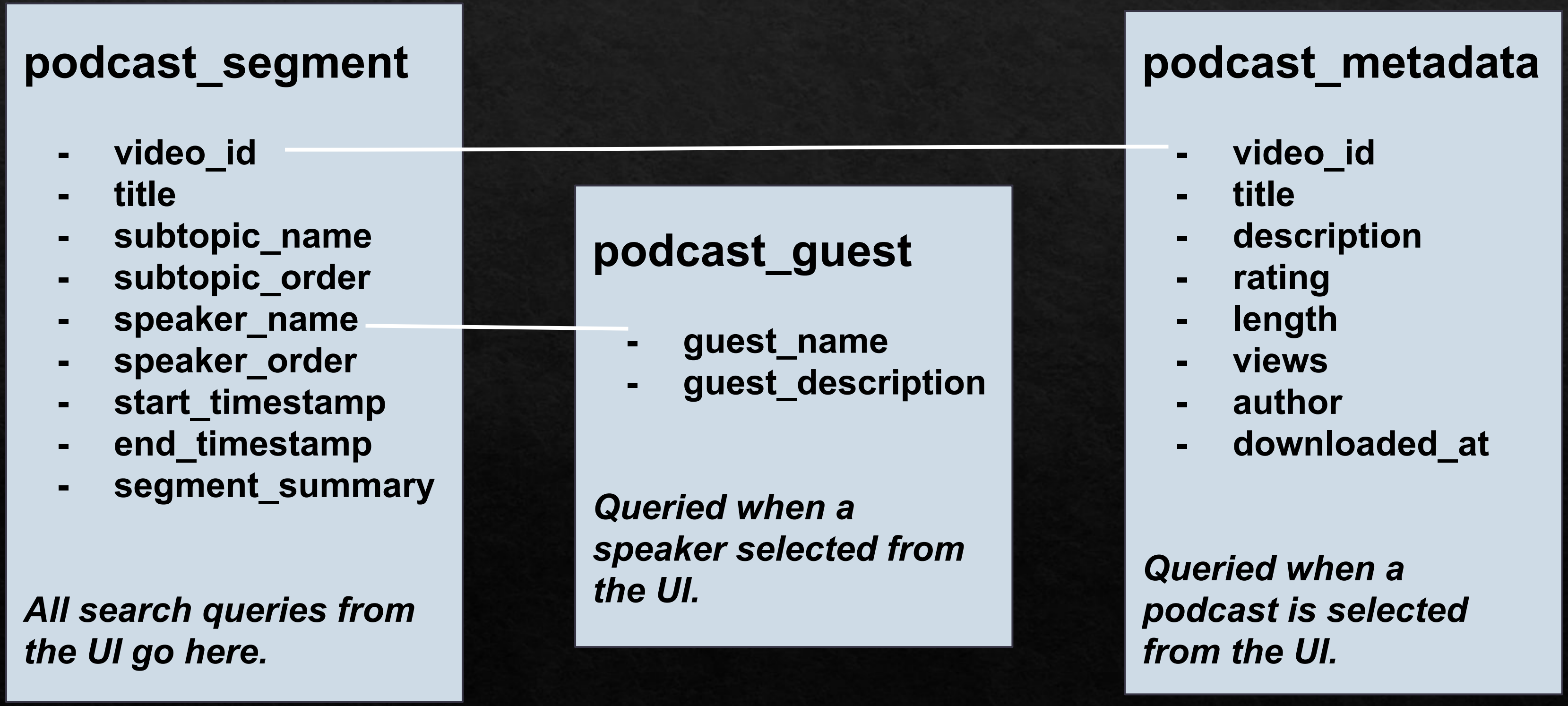
Raw data schema - MongoDB

segment

- **video_id**
- **title**
- **subtopic_name**
- **subtopic_order**
- **speaker_name**
- **speaker_order**
- **start_timestamp**
- **end_timestamp**
- **raw_text**

No queries from the end-user go here. This is a hidden archive of raw data.

Processed Data Model - Elasticsearch



podcast_segment

- video_id
- title
- subtopic_name
- subtopic_order
- speaker_name
- speaker_order
- start_timestamp
- end_timestamp
- segment_summary

All search queries from the UI go here.

podcast_guest

- guest_name
- guest_description

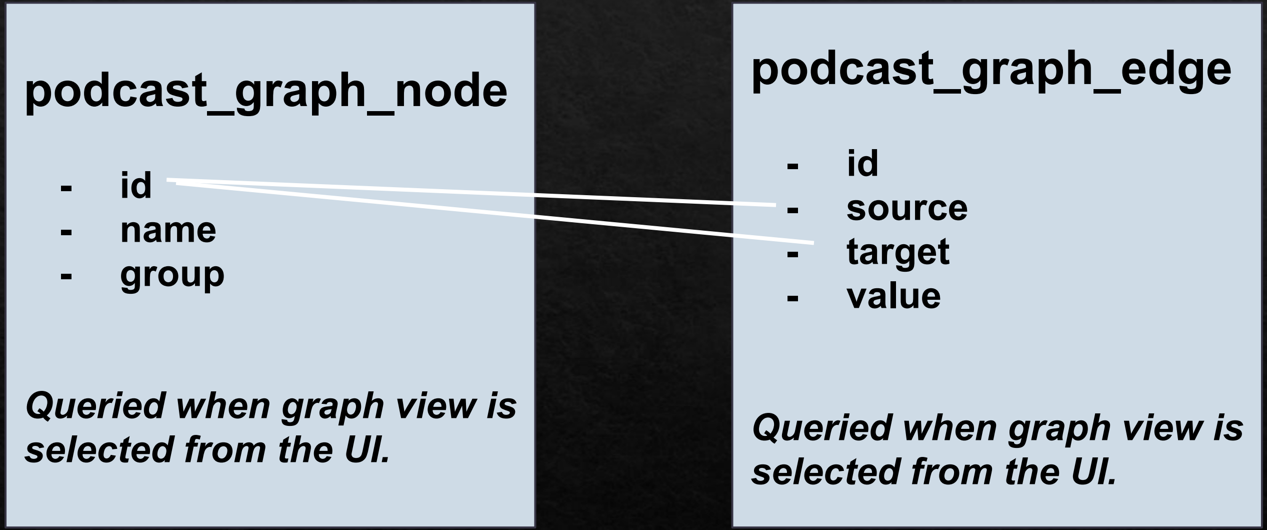
Queried when a speaker selected from the UI.

podcast_metadata

- video_id
- title
- description
- rating
- length
- views
- author
- downloaded_at

Queried when a podcast is selected from the UI.

Knowledge Graph Data Model - Elasticsearch



podcast_graph_node

- **id**
- **name**
- **group**

Queried when graph view is selected from the UI.

podcast_graph_edge

- **id**
- **source**
- **target**
- **value**

Queried when graph view is selected from the UI.

Challenges

CURRENT

- Highly unstructured speech to text output in terms of grammatical syntax (for example, no punctuation), leading to lots of garbage entities in knowledge graph

- Pretrained models for speaker diarization, speech to text, text summarization and named entity recognition not too accurate, need to build better models through transfer learning on specific dataset

FUTURE

- Traditional approaches to graph visualization will fail as the knowledge graph keeps growing in size with more data

- Increasing readability in UI design and improving using experience