
Knowledge
Extraction from
Podcasts

Dataset Overview

[Lex Fridman Podcast playlist on Youtube]

- 80 episodes with timestamps in description (as on 12/15/2020), each around 2-4 hours.
- Dynamic data, with 2 to 3 new podcasts added every week.

Unstructured raw data
- Audio files

Semi - Structured metadata & transcripts
- 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
- But as opposed to mainstream media, contain invaluable information, expert opinions and diverse perspectives, ranging from science, society, politics, to life and beyond
- By efficiently wrangling large and unstructured podcast audios, we can create an easy to explore knowledge repository for users and researchers

Fundamental Features

- **Easy to use search interface to discover podcasts of interest**
  Input - Query (string);
  Output - List of relevant podcasts, segments, guests relevant to the query

- **Podcasts are long and do not discuss just one topic throughout the episode. Users can specifically listen to the segments they are interested in**

  Input - Play/stop segment audio button; Output - segment audio

- **Get a complete view of the various entities talked about in podcasts**
  Input - mouse clicks/selections; Output - graph view

Current scope of project

YES
NO

For example, Google

USER INTERFACE

QUERY

RESPONSE

REAL TIME PROCESS

SEARCH INDEX STORE

GOAL : enhance the search experience as much as possible

RESEARCHER

EXPERIMENTATION ON MASSIVE KNOWLEDGE GRAPH

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

BACKGROUND PROCESSES

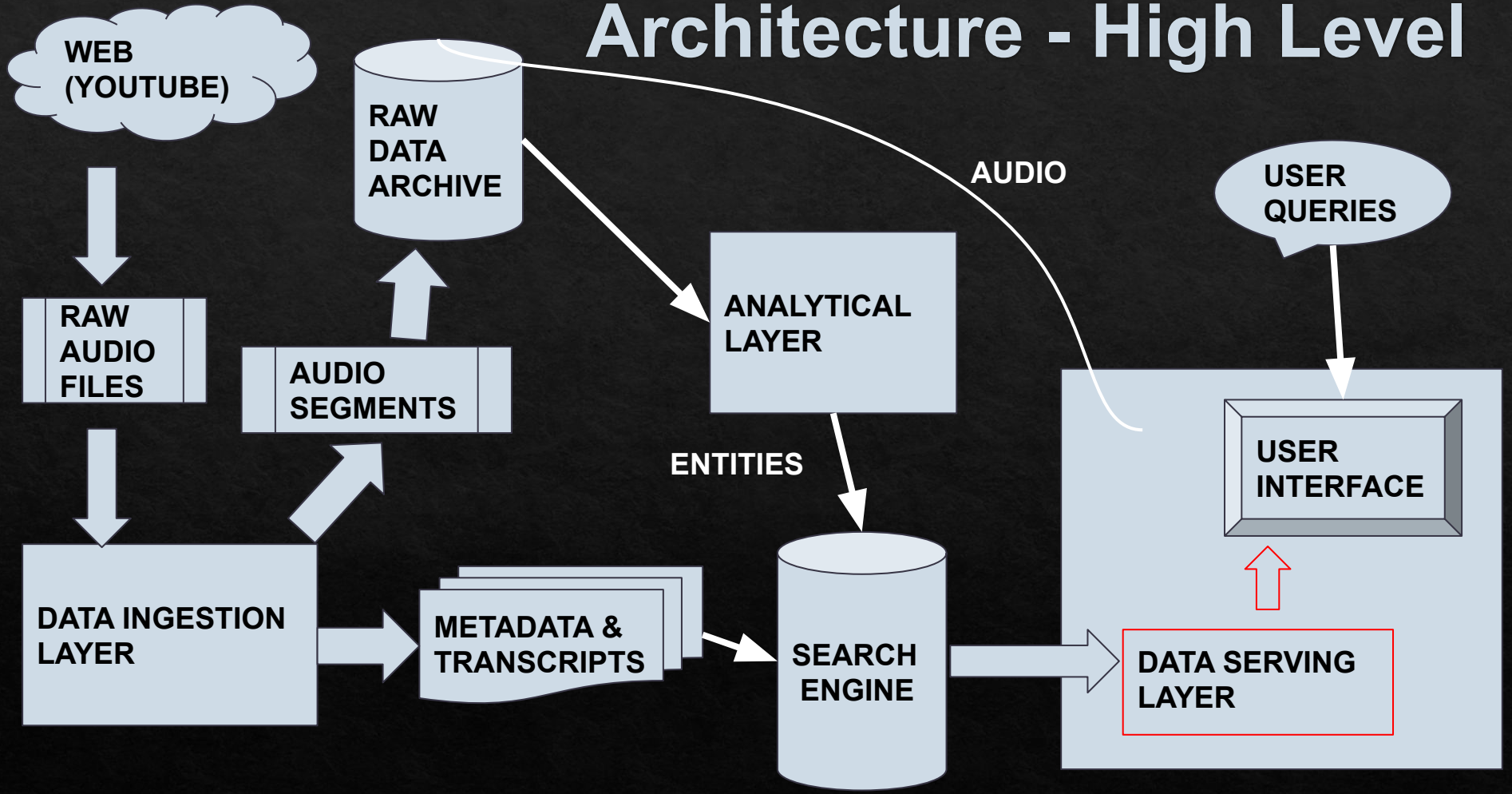
KNOWLEDGE GRAPH

Updates for improved search results or recommendations

Types of End users

- Naive Users - UI to explore podcasts, segments, guests of interest
- Technical CS User - End to end data pipeline to ingest, transform and serve audio datasets
- Researcher - Leverage entities/knowledge graphs to find new links

Architecture - High Level



Architecture - End to End

DATA INGESTION LAYER

- **EXTRACT**
  - ***Data Downloader*** **: downloads a Youtube video as a .wav audio file and also creates metadata from the video description as a text file**

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

  - ***Speaker Diarization*** **: partition each subtopic audio into timestamps according to the 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 smaller speaker segments**

- **LOAD**
  - ***Raw Data Load*** **: loads each smallest audio segment to raw data store and creates a corresponding metadata record for it in raw data store**

  - ***Transcribe*** **: finds untranscribed audio segments in the raw data store, converts them to text and adds transcriptions to their corresponding metadata in the raw data store**

Architecture - End to End

DATA INGESTION LAYER

- **LOAD**
  - ***Index Segments*** **: reads transcripts of audio segments from the raw data store and indexes them on the search engine**
  - ***Index Metadata and Speakers*** **: reads text metadata files for each podcast and indexes speakers and podcast details on the search engine**

ANALYTICAL LAYER

Note **: Since the transcription done by Transcribe (LOAD) module is not very accurate, we use pre-compiled transcripts downloaded from the web (happyscibe) for entities extraction.**

- ***Entities extraction*** **: extracts key entities for each podcast and indexes them on the search engine**

Architecture - End to End

DATA SERVING LAYER (Application and User Interface)

- User interface to the end-user for searching the knowledge repository built over podcasts data
- Provides REST endpoints to retrieve processed data from search engine or raw data from the raw data store
- Visualizes extracted entities as force directed graphs

Tools & Libraries

- Programming Languages
  - Backend : Python and Java (Multithreading)
  - Frontend : HTML and JavaScript
- Data Lake
  - Raw data archive : MongoDB & GridFS
  - Search engine : Elasticsearch
- Data Download (Data Ingestion Layer)
  - pytube + ffmpeg
- Audio Segmentation (Data Ingestion Layer)
  - pydub

Tools & Libraries

- Named Entity Recognition (Analytical Layer)
  - Spacy
- Speaker Diarization (Data Ingestion Layer)
  - CMU Sphinx
- Audio to text transcription (Data Ingestion Layer)
  - Mozilla deepspeech
- Web application (Data Serving Layer)
  - flask
- Graph visualization (Data Serving Layer)
  - D3.js (force directed graph)

Raw data schema - MongoDB

segment

1. Video_id
2. Title
3. Subtopic_name
4. Subtopic_order
5. Speaker_name
6. Speaker_order
7. Start_timestamp
8. End_timestamp
9. segment_transcript

About Podcast

segment

10. Gridfs_key
11. Nchannels
12. Sampwidth
13. Framerate
14. Nframes
15. Comptype
16. Compname

About Audio Files in GridFS

Processed Data Model - Elasticsearch

podcast_segment

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

All search queries from the UI go here.

podcast_guest

- guest_name
- guest_description

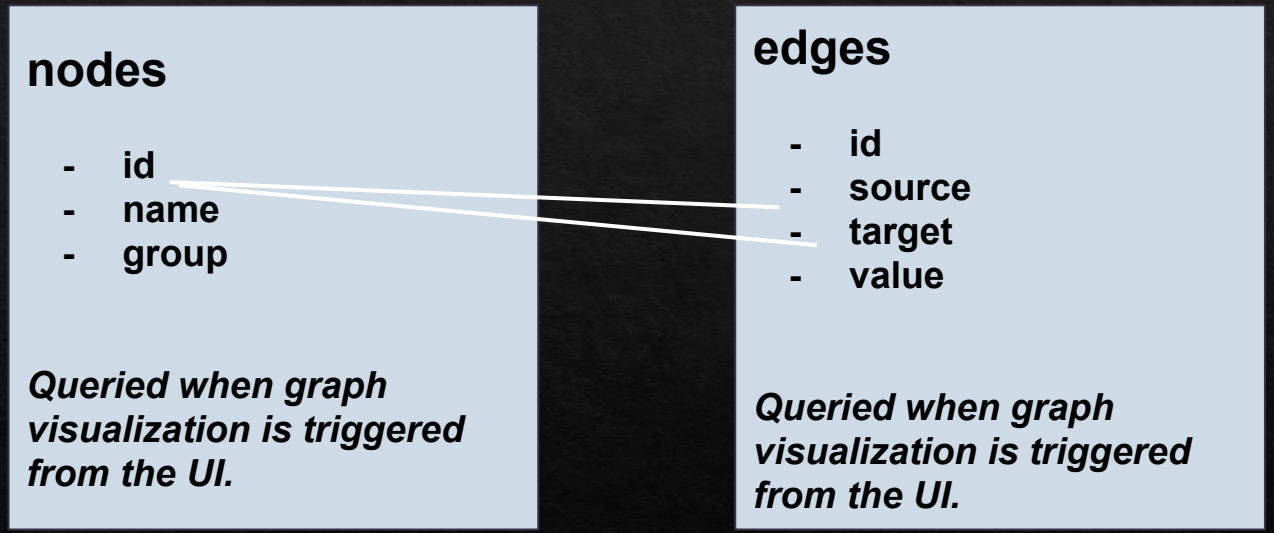
Queried when a speaker selected from the UI.

podcast_metadata

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

Queried when a podcast is selected from the UI.

Graph Data Model - Elasticsearch



Challenges

CURRENT

- Speech recognition is hard due to lot of variation in human voice and tone. As a result, speech-to-text transcription is not very accurate. For entity extraction, we used pre-compiled transcripts from happyscribe.com

- Major reliance on pre-trained models and libraries leading to limited scope for customization.

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 user experience.