Technical Summary

SubtitleVision aims to bring subtitles off your TV and onto your phone. We are not only an accessibility aid but also a companion. We aim to achieve this by designing a comfortable iOS application. We are building an iOS app because our target audience is college-aged people and young professionals who primarily use iPhones. Most of the computation will happen on the backend. A flask server will act as the endpoint for the iOS application to interact with. Flask is a simple and flexible web server framework for python our main server-side language.

The iOS application will send an audio recording to the server, which will be processed using Microsoft Azure Cognitive Services. This will convert the speech to text. It will attempt to convert a small sample to text and search against that. If it cannot find a match it will get a longer sample and continue to search until a match is found. We have set up a baseline iOS application that can record a sample and send a request to Microsoft Cognitive Services to transcribe the audio.

Given the speech-to-text may not be completely accurate, a fuzzy search will be performed using the text generated and a database of various movie and TV show subtitle files. The search algorithm will present some challenges. Though text search is a well-documented problem so it should be doable.

To get the database, we will be using OpenSubtitle's API to download subtitle files, TMDb to get metadata, and IMSDb to get screenplay information. Downloading and parsing subtitle files is relatively easy. Getting the information on which subtitles to download has been a bit more difficult. We will scrape the internet for popular movies and TV shows to pass into the application that downloads the subtitles. We will also use web scraping to get information from IMSDb. Since IMSDb holds screenplays in HTML rather than PDF like other screenplay databases, this should allow us to easily get character information.

After it finds the subtitle file corresponding to the input recording, we will synchronize the subtitles using a method called forced alignment. Forced alignment is the process in which you take orthographic transcription and align it with audio. This presents some challenges however, we expect it is doable. There are some pre-trained models for forced alignment to guide us in our algorithm.

One of the main costs is the server hosting the backend of this project. We don't expect it to need very much computational power and we're only aiming for 1GB of storage for our file database. We are hoping to accomplish a naive version of the search and alignment with a limited dataset of subtitle files by December to have a rudimentary implementation of the app. At the beginning of next year, we hope to expand our repository of subtitles as well as collect data from screenplays. By the second quarter of next year, we are aiming to have refined algorithms and a polished design for the app.

Product Specifications

User Stories

As a movie watcher, I would like to have subtitles on my phone, so that I can understand what the characters are saying without disrupting other people.

As a movie watcher, I would like to have translations, so that I can watch a movie in my non-native language.

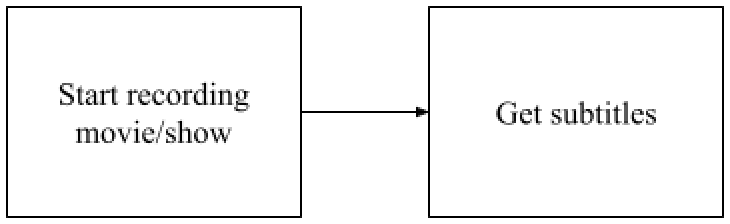
As a parent, I would like to have content warnings, so that I can skip sensitive scenes in movies.

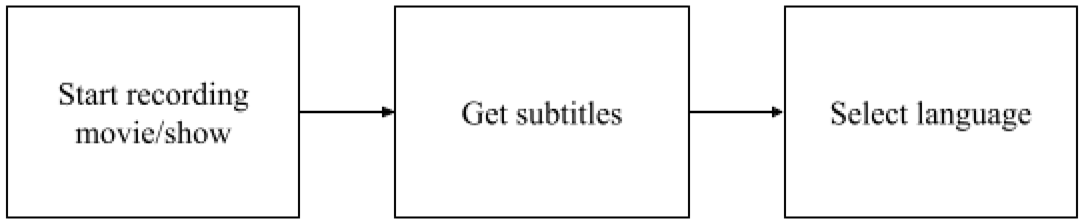
As a movie watcher, I want to view timely, tangential information about films such as who is speaking, what actors are in a scene, and what music is playing.

As a movie watcher, I want to be able to quickly go back and view lines that were previously said if I missed a section in the content without rewinding.

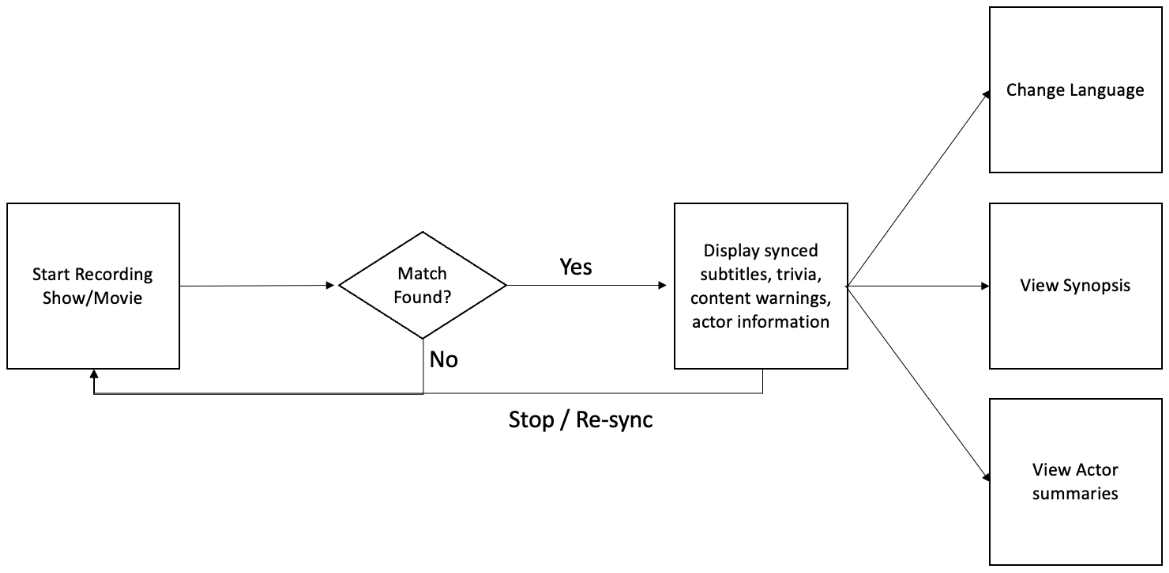
Flow Diagrams

High-Level Flow:

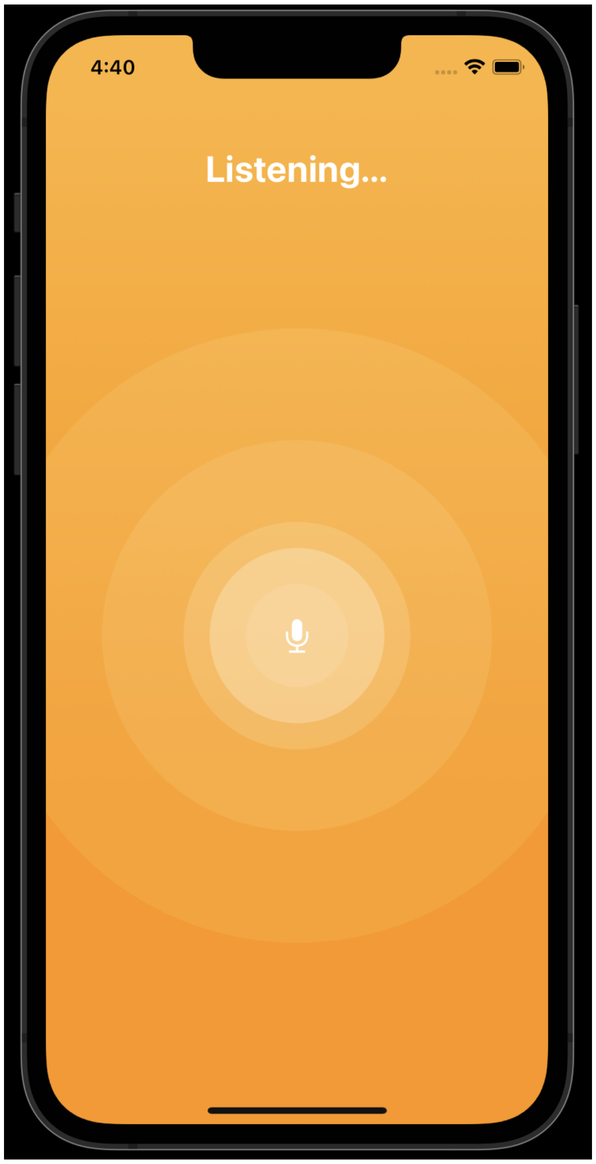


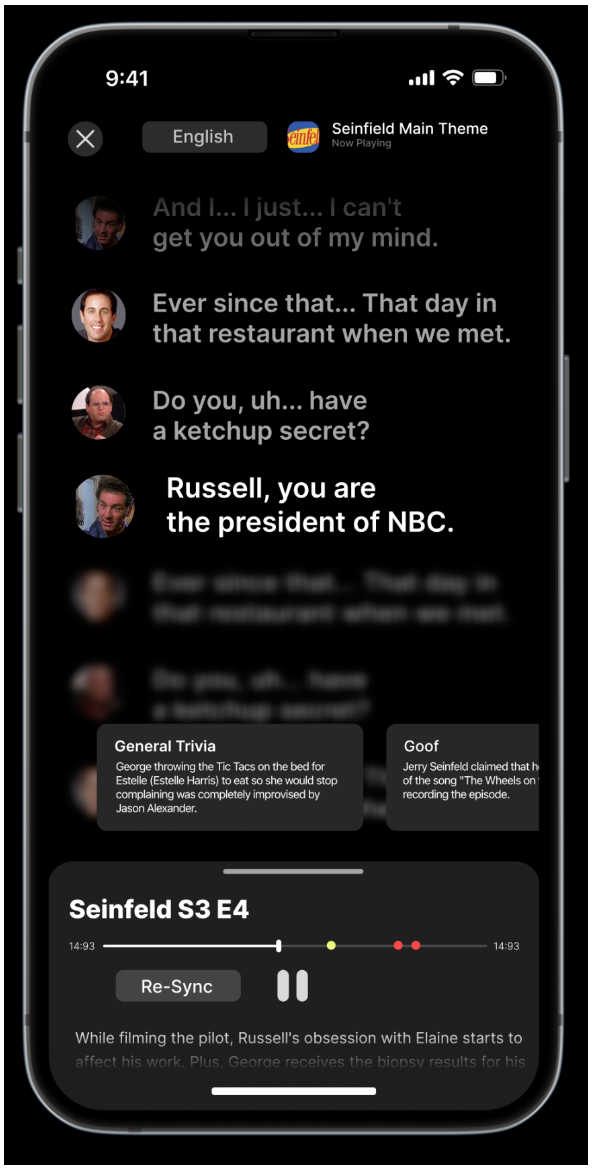


Detailed Flow:


Start Recording Show/Movie
Match Found?
Yes
No
Display synced subtitles, trivia, content warnings, actor information
Stop / Re-sync
Change Language
View Synopsis
View Actor summaries

Mockup




9:41

English

Seinfield Main Theme
Now Playing

And I... I just... I can't get you out of my mind.

Ever since that... That day in that restaurant when we met.

Do you, uh... have a ketchup secret?

Russell, you are the president of NBC.

Ever since that... That day in that restaurant when we met.

Do you, uh... have a ketchup secret?

General Trivia
George throwing the Tic Tacs on the bed for Estelle (Estelle Harris) to eat so she would stop complaining was completely improvised by Jason Alexander.

Goof
Jerry Seinfeld claimed that h
of the song "The Wheels on
recording the episode.

Seinfeld S3 E4

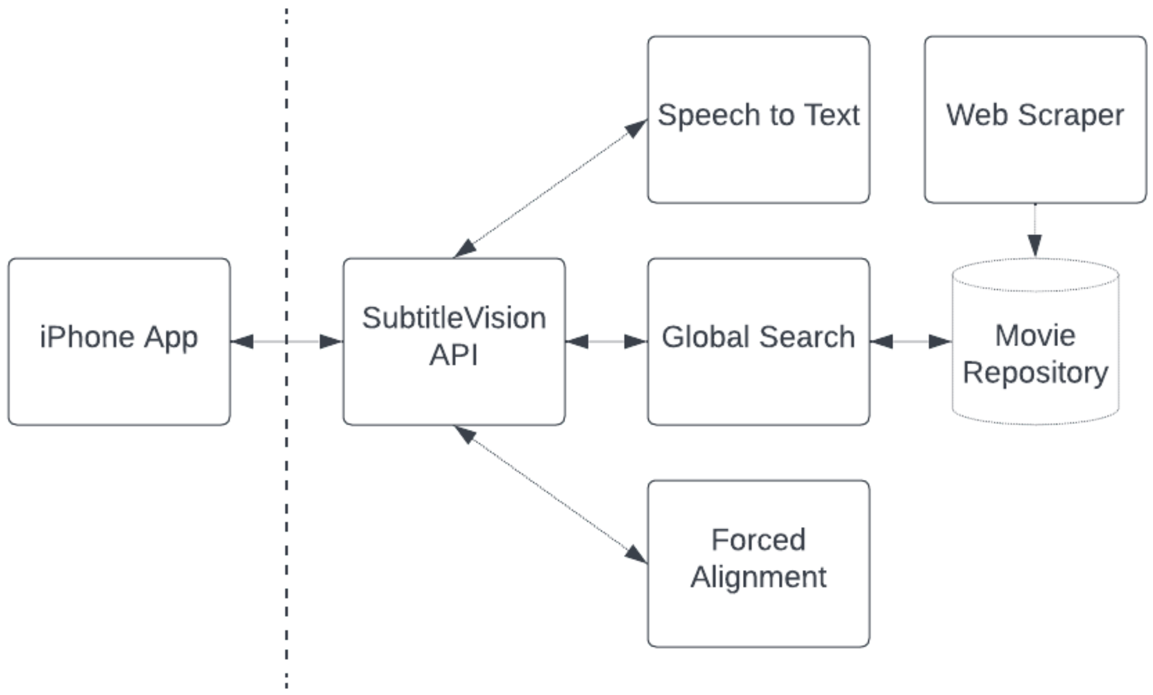
14:93 14:93

Re-Sync

While filming the pilot, Russell's obsession with Elaine starts to affect his work. Plus, George receives the biopsy results for his

Technical Specifications

System Architecture



SubtitleVision API Specification:
This is the specification for the Flask API that will send data to the iOS app.

`POST /recording`

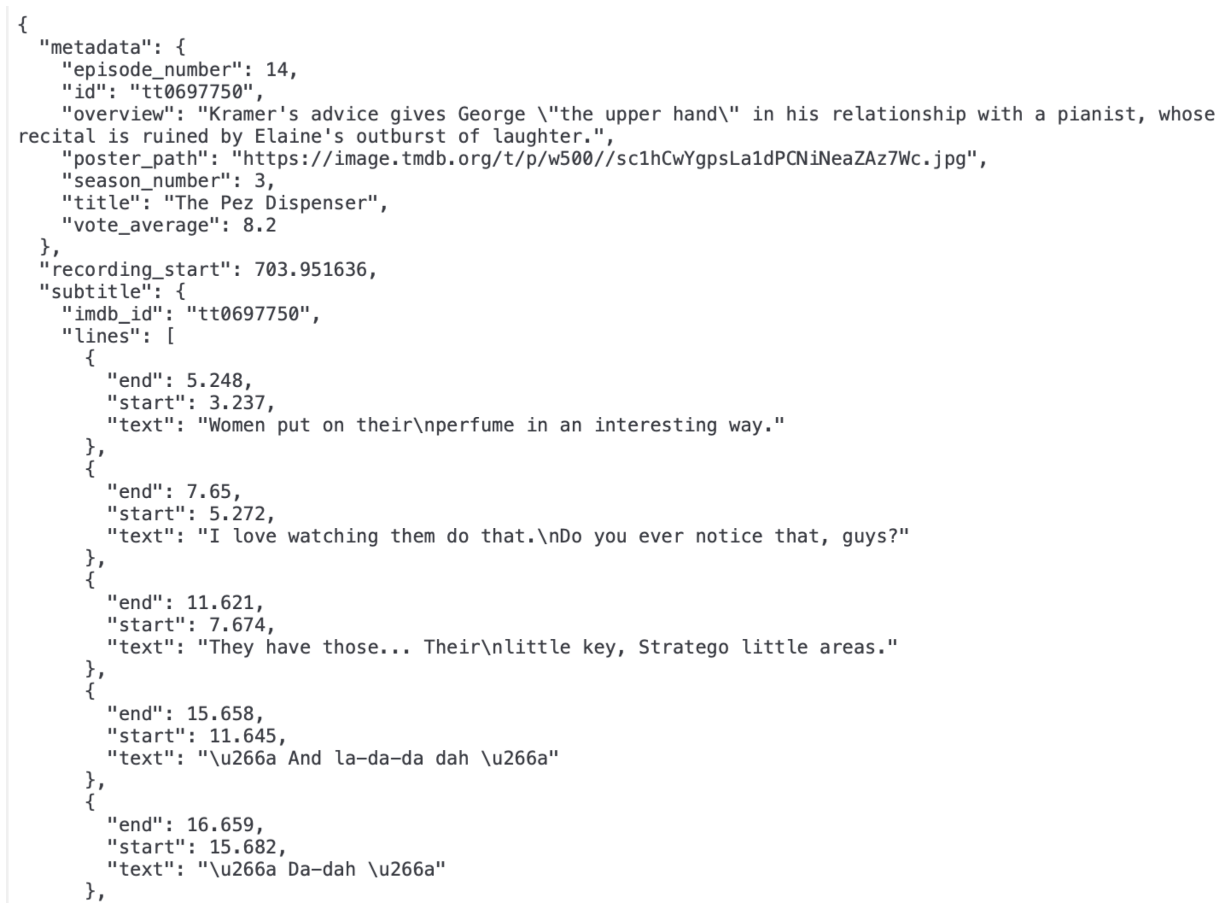
Creates a new recording resource and returns id

Example Response:
```
{ recording_id: "1234567"}
```

`PUT /recording/<recording_id>`

Appends given .wav file to recording with the given id, if there is a match it returns the movie information and subtitles

Parameters: `recording_id`, *Multipart .wav file*
Example Response:



```
{
  "metadata": {
    "episode_number": 14,
    "id": "tt0697750",
    "overview": "Kramer's advice gives George \"the upper hand\" in his relationship with a pianist, whose
recital is ruined by Elaine's outburst of laughter.",
    "poster_path": "https://image.tmdb.org/t/p/w500//sc1hCwYgpsLa1dPCNiNeaZAz7Wc.jpg",
    "season_number": 3,
    "title": "The Pez Dispenser",
    "vote_average": 8.2
  },
  "recording_start": 703.951636,
  "subtitle": {
    "imdb_id": "tt0697750",
    "lines": [
      {
        "end": 5.248,
        "start": 3.237,
        "text": "Women put on their\nperfume in an interesting way."
      },
      {
        "end": 7.65,
        "start": 5.272,
        "text": "I love watching them do that.\nDo you ever notice that, guys?"
      },
      {
        "end": 11.621,
        "start": 7.674,
        "text": "They have those... Their\nlittle key, Stratego little areas."
      },
      {
        "end": 15.658,
        "start": 11.645,
        "text": "\u266a And la-da-da dah \u266a"
      },
      {
        "end": 16.659,
        "start": 15.682,
        "text": "\u266a Da-dah \u266a"
      },
```

…

`DELETE /recording/<recording_id>`

Deletes the specified recording resource

Parameters: `recording_id`

External APIs and Frameworks

Swift + SwiftUI

Goal

- A display that is convenient and intuitive for Movies, TV Shows, and Subtitles

Description

We use Apple's SwiftUI framework to create an intuitive, visually pleasing interface for viewing timely movie information on an iPhone.

Python Flask

Goal

- Utilize speech-to-text, global search, and subtitle aligner services to provide necessary information back to the client

Description

Flask is used to create a RESTful web API for the iOS app to interact with our server components in order to obtain movie information and time-synchronization information. The code for our endpoints calls speech to text, global search, and subtitle aligner services, and synthesizes the information into the JSON format for the iOS app to use.

Azure Cognitive Services

Goal

- Transcribe the audio sample to text

Description

This API is used in the speech_recgonition file in the code. It gets called while the phone is recording audio. It takes in the ID of a recording sample and returns a transcription of the audio.

FuzzyWuzzy

Goal

- Determine the playing movie or TV show and the approximate playback time

Description

This library is used in the moviematch file in the code for "fuzzy" search. It uses Levenshtein Distance to calculate a score for how closely two lines of text match. It gets called after a transcription is generated from the cognitive services API in order to find the closest matching line from all subtitles in the repository.

PyTorch/TorchAudio

Goal

- Match a time in the speech sample with an exact playback time in the subtitle file

Description

This library is used in the alignment file in the code. It gets called after a transcription is found from the search. It takes in a recording ID and a transcription and returns the start time of the last word in the audio recording.

Beautiful Soup

Goal

- Gather data from the internet regarding various shows or movies

Description

This library is used in the imdbScraper file in code. It is used to gather information from websites by parsing the page into searchable elements that can be stored for later.

OpenSubtitles

Goal

- Create a repository of movie and show subtitle files

Description

This library is used in the openSubtitles file in code. It is called in the subDownload file. It searches for a subtitle given an IMDb title ID and a language. It then downloads the subtitle into a .srt file.

Endpoints Used

- POST /api/v1/login
- GET /api/v1/infos/user
- GET /api/v1/subtitles
- POST /api/v1/download

Pysrt

Goal

- Parse subrip subtitle (srt) file into useable JSON for iOS app

Description

This library is used to parse the subtitle file (srt) in code. It is used to open srt files, and convert them to usable json.

TMDb API

Goal

- Gather metadata for a movie or show from IMDb ID

Description

This API is used in the moviemetadata file in code. It takes in the IMDb ID of a movie or show and returns information such as title, overview, poster, rating, season, and episode number if applicable.

SQLAlchemy + SQLite

Goal

- Keep track of where recordings from phone are stored in order to manage, merge, and delete multiple recording segments

Description

SQLAlchemy, together with a standard SQLite database, allow us to easily track the file location of recordings within our Python Flask web API and delete them after processing. When the client app sends a recording, it will be saved to a static folder and tracked within a SQLite table along with metadata such as the start date and client id. SQLAlchemy serves as an ORM to simplify interactions with the database.

Algorithms

Global Search

Goal

Search for matching subtitle files from a given transcription line.

Description

The goal of global search is to take a transcription of audio and determine: which movie it is from and what line of text within that movie it is. We do this using a combination of natural language processing (NLP) and fuzzy text search. NLP is used as a performance measure. Essentially using a hashmap we search the nouns found in the transcription against all the nouns in every movie we support. Once we have candidate movies from NLP we perform a costly fuzzy search against the transcription and every line in the candidates. This gets the exact movie and the line within the movie.

Forced Alignment

Goal

Align recording file with transcription

Description

From global search, the input of our subtitle synchronization service will be 3-5 lines of subtitle information along with the 15-30 second recording from a user's phone. This information must be synthesized in order to produce multiple "anchor points": time codes within the movie and their corresponding time code within the audio recording. Using torchaudio's Wav2Vec2 pre-trained speech-to-text machine learning model, we will generate frame-wise label probabilities of the audio and then generate a matrix containing the probability that a transcript label will occur at each time frame. From there we will traverse the matrix following the elements with the highest probabilities. Then we will merge repetitions with the same label by taking the average of the merged segments. Finally, words will be merged together and the start time of the final word of the audio recording will be returned so that we can synchronize the subtitles with the movie or show being played. Because the subtitle transcripts include line-by-line time codes, to produce anchor points, we simply need to pick the first word in the subtitle lines–the time they are said is now known as well as their time within the content.