RIME-833 : SEMESTER PROJECT REPORT **Group 10**

Problem Statement

The goal of the project is to recognize the brand names being advertised in a series of varying advertisements provided as a video input.

Workflow

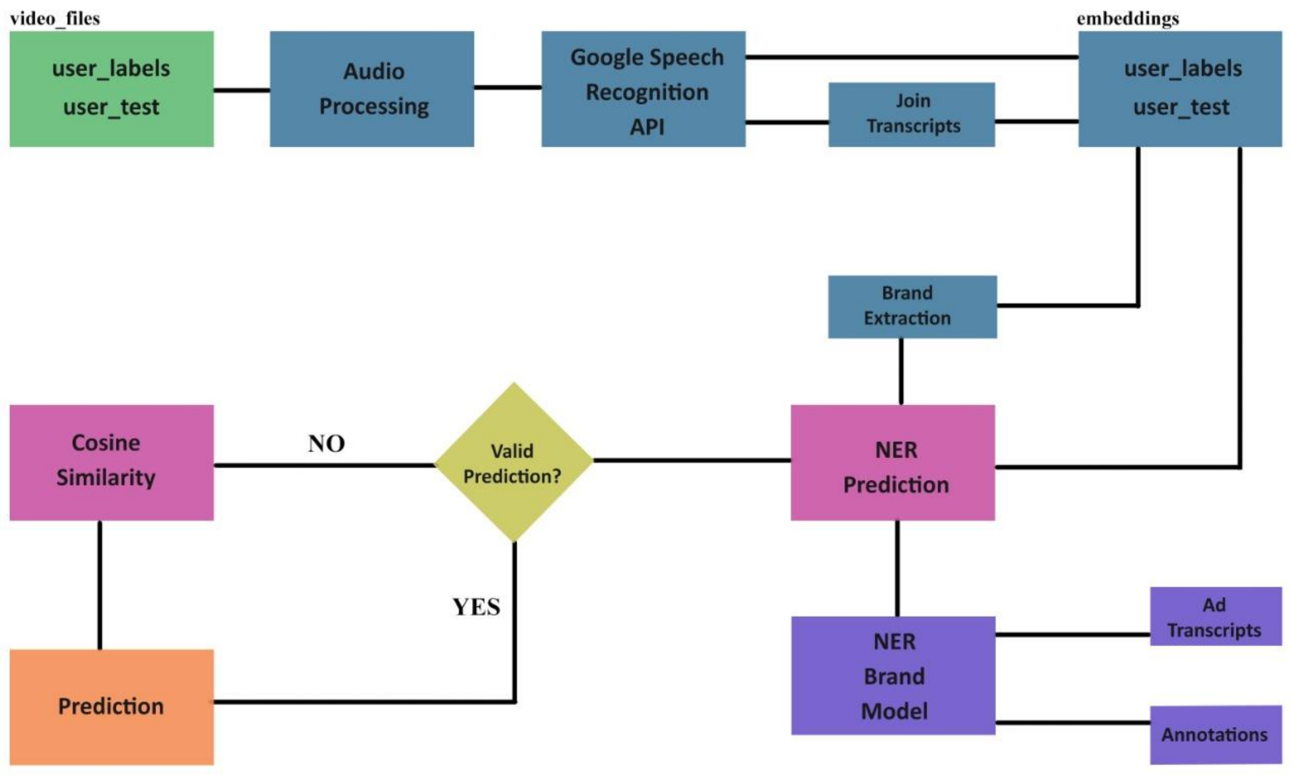
Prior to runtime, a user provides a number of sample individual video advertisements with their label brand in the model repository and a test video consisting of one or more commercials each advertising any one of the sample brands.

The sample advertisements are added to the *video_files/user_labels* directory with filenames as brand labels and the test video is added to the *video_files/user_test* directory for inference. Filenames should contain **no spaces**.

The *main.py* file is executed for processing and prediction.

Project Pipeline

The following pipeline is applied for brand prediction using the sample advertisements on the test video:

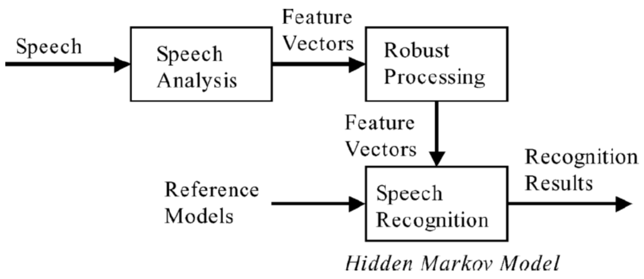


Preprocessing

The labelled advertisements are preprocessed to create a reference embedding associated to the brand for each video file. The test video is similarly processed to produce a test embedding for prediction.

i. **Audio Extraction**
Audio data from each video is extracted using *FFmpeg* in the shell and saved as *audio_mp3.mp3* in the *audio_cahe/* directory. The *.mp3* file is then converted to a *.wav* file and stored in the same directory and split into smaller chunks of *.wav* files stored at *audio_cache/splits/*. The chunk size for labelled videos is set at 15 seconds and for the test video at 20 seconds. The files are overwritten for each video processed.

ii. **Speech to Text Conversion**
The *Google Speech Recognition API* provided in the *SpeechRecognition* package is used to produce text transcripts using the split *.wav* files for each labelled video and the test video. The transcripts generated from audio chunks of a labelled video are concatenated as a single text variable but the transcripts generated from audio chunks of the test video are treated separate. This enables a running prediction on the test video at the prediction stage. The *Google Speech Recognition API* uses a Hidden Markov Model on extracted feature vectors from the audio speech to provide English transcription for audio samples in English as well as Urdu language. The API is also the major cause of model latency for prediction. The following figure provides a brief schematic of the *Google Speech Recognition API* architecture.



iii. **Text Processing**
The text from the transcripts is processed to create an embedding from each labelled video and chunks of the test video. The *Natural Language Tool Kit (NLTK)* is utilized for most parts of text processing. The text is tokenized and filtered to eliminate any stop words from the word list. The filtered word list undergoes Part of Speech (POS) tagging to extract the Nouns, Verbs and Adjectives from the word list. The tagged words are grouped by frequency into a *.json* file and stored as feature embedding of each video. Each embedding holds the text used to generate it. The embedding generated from a labelled video is stored at *embeddings/user_label/* directory and the embeddings generated the test video is stored at *embeddings/user_test/* directory.

Prediction

The embeddings from the labelled videos and the test video are used in a multi-step approach the predict the brand/s advertised in the test video.

i. **Named Entity Recognition (NER)**

The Named Entity Recognition model from *Spacy* was used and custom trained to recognize possible brand names present within written sentences. The dataset used for training consisted of more than 1000 ad transcripts annotated over 700 different brands to recognize any brand name possibly present in a provided statement. The advantage of using a NER model for brand name recognition is that it enables the possibility to reduce prediction inaccuracies arriving from faulty embeddings produced from failed transcriptions.

The embeddings from the labelled videos are used to produce a subspace of possible labels for prediction. The parent text from each test embedding is passed to the NER model for brand name detection and any brand detected is searched for in the labelled brands subspace. A match for a detected brand using this approach ensures a near 100% accuracy and is returned as prediction.

ii. **Cosine Similarity**

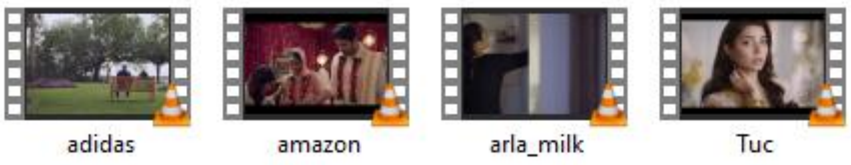
In case the NER model fails to detect any brand in a test video embedding, the embedding is compared with each of the labelled video embedding to calculate the *Cosine Similarity score* of each comparison. The *Cosine Similarity score* is calculated by creating feature vectors from both the reference and test embedding where a feature vector is a n-dimensional vector representing the frequency of each word in its corresponding embedding from the set of words present in both embeddings. The score is calculated is as:

$$sim(\boldsymbol{v_1}, \boldsymbol{v_2}) = \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{||\boldsymbol{v_1}||\ ||\boldsymbol{v_2}||}$$

The brand associated with the reference embedding with the highest score is returned as prediction.

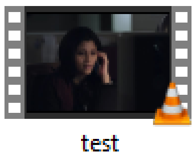
Example Run

The following advertisement videos were provided as sample labels in the *video_files/user_labels* directory:

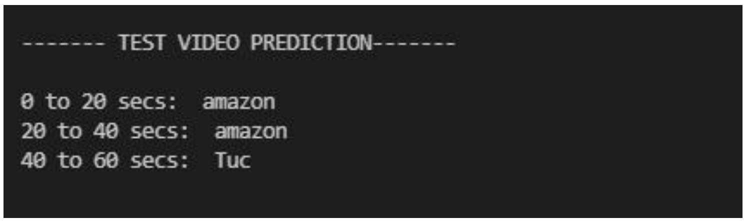


This creates a brand subspace of ***Adidas***, ***Amazon***, ***Arla Milk*** and ***Tuc***.

Then a test video which contained an advertisement for Adidas was added to *video_files/user_test* directory:



Executing the *main.py* file generated the following prediction:



```
------- TEST VIDEO PREDICTION-------

0 to 20 secs:  amazon
20 to 40 secs:  amazon
40 to 60 secs:  Tuc
```

The model provided correct prediction for the first 40 seconds of the test advertisement!

Limitations

1. Google Speech Recognition API increases latency
2. Noise in audio reduces transcription accuracy
3. Only audio data used for prediction. No video features utilized.

Attachments

1. Report
2. Code Files
3. Code PDFs