



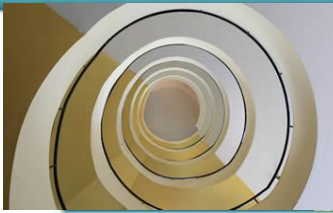
Intel - Final Presentation

Project Title : Searching a Video Database using Natural Language Queries

Project Guide : Dr. Mamatha H R

Project Team : Shubha M (PES1201701540),
Shrutiya M (PES1201700160),
Kritika Kapoor (PES1201701868)





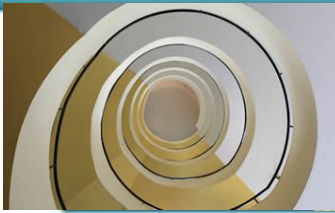
Project Abstract and Scope

SEARCHING A VIDEO DATABASE:

An application that achieves voice based **natural language query**, search and extracted video segment playing after the search in order to query the content of the videos in a user-friendly manner is built.

The scope of querying video databases is huge. For example, consider the following queries:

1. A red car in front of a white building. (This can be queried in security footage database)
2. Man in a blue jacket next to a woman. (This sort of queries can be used in journalism for identification)
3. Ball in goal post. (This can be queried in sports events video databases)
4. College lecture videos. By giving a keyword as the query, we can retrieve the video segments of a huge video where that or anything similar to that is being explained.



Design Approach

We have two design approaches , one considering the visual features and the the other considering audio features of the video.

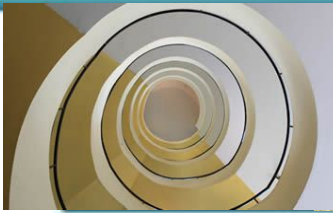
Visual Approach:

The architecture is designed using two different image captioning models Densecap and NeuralTalk2. It consists mainly of 3 sequential parts:

- i. Generate Captions
- ii. Creating Tracklets
- iii. Voice based Search.

1. Generating Captions

- Splitting the video clip into frames.
- After this, the image captioning models were applied on each frame.
- NeuralTalk2 generated one caption whereas Densecap generated multiple captions corresponding to specific regions of interest, denoted by bounding boxes, in the image.



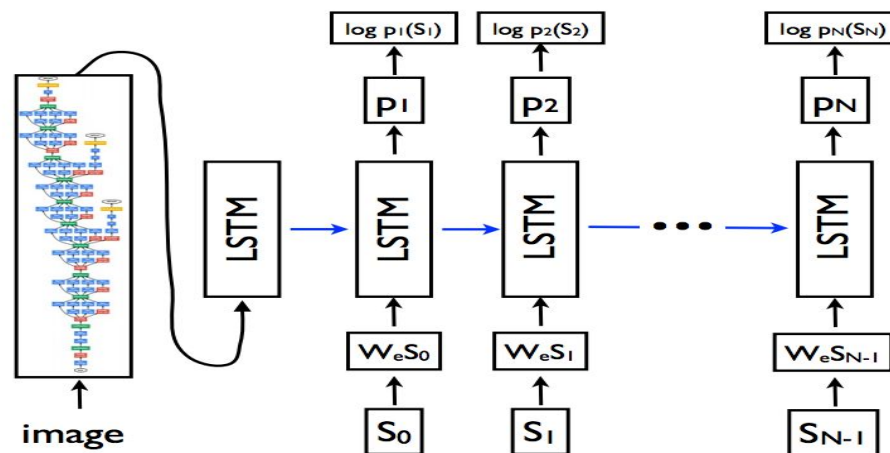
Design Approach

2. Creating Tracklets

The creation of tracklets vary for the NeuralTalk2 and Densecap approach.

a. NeuralTalk2 -

- The first tracklet is represented by the caption of the first image and the rest of the frames are sequentially compared with the tracklet upto the previous frame.
- The criteria of inclusion of a frame is the semantic similarity exceeding the threshold.
- Completion of tracklets is achieved by the number of frames calculated to be dissimilar to the tracklet caption exceeds the cutting threshold.

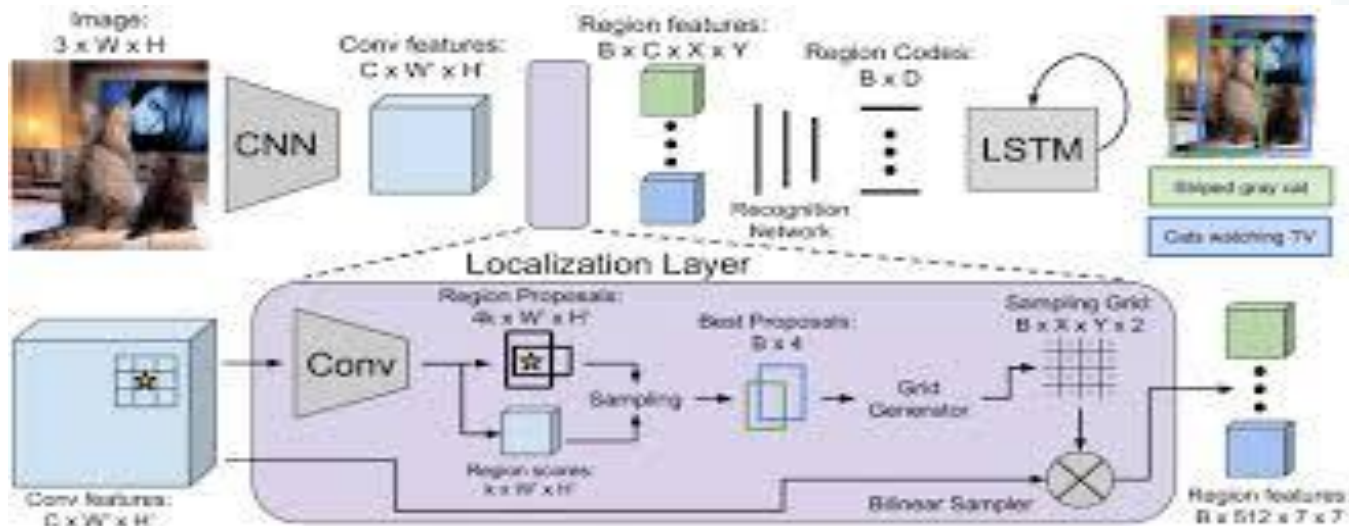


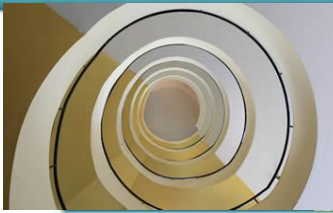


Design Approach

b. Denscap -

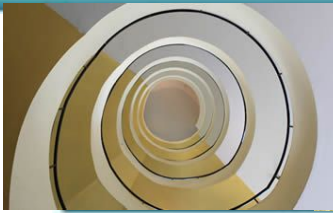
- It outputs a json file, with 80 captions(avg.) per image mapped with the corresponding weights, and bounding boxes of the objects.
- If frame has N captions, N tracklets are begun each represented by these captions. A new frame is compared, and if M captions are found to be similar, the unmatched N-M captions are registered as new tracklets, resulting in the total of 2N-M tracklets.
- First 5 highest weight captions are used.
- **Semantic similarity threshold** and **cutting threshold** concept is used here too to continue the process.





Design Approach

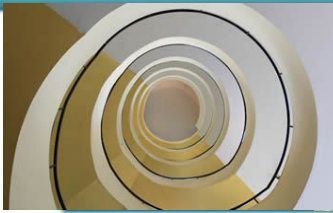
- The euclidean distance of the bounding boxes of two frames is used to decide spatial similarity.
- This generates a set of tracklets each represented by the caption of the first frame, and each tracklet storing information of the frames it constitutes.
- Given an input video, it is processed upon by both of these methods and broken down into a set of tracklets.
- After this the duration of every tracklet was calculated by mapping it to the input video duration and this data was stored in a file.



Design Approach

3. Voice based Query

- The input voice query is converted to text using a speech recognition system
- The query is compared with every representative caption of the final tracklets and outputs the tracks that are semantically relevant.
- Google's Universal sentence-encoder is employed. Using a word based embedding approach and obtaining average of word embedding will mostly not represent the actual meaning of a sentence.
- Finally, cosine similarity metric is used as the similarity measure.



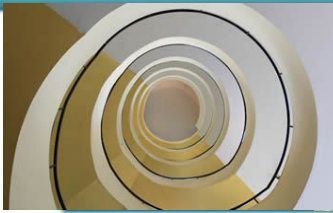
Design Approach

Audio Approach:

- As mentioned in the upcoming plans of the 90% milestone submission section, we had planned to process the audio of the video to create tracks based on the input query.

Scope and Uses of this approach:

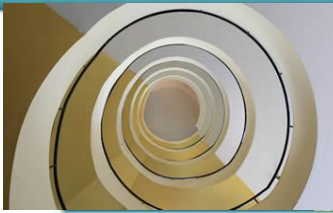
- This acts as an added approach to creating tracklets as audio might be important in certain situations and can provide vital information to tracklet formation.
- This can be used on all videos with audio to seek to the part where anything related to the given query was present (Semantically similar).
- A very useful example would be a College lecture video. By giving a keyword as the query, we can retrieve the video segments of a huge video where that or anything similar to that is being explained.



Design Approach

Procedure:

- For this approach, we process the transcripts generated by YouTube for a particular video and first store all the transcripts with their respective start and end timings
- The voice query is taken as input and the most relevant segments of the video are retrieved using a on-the-fly generation of tracks. A cutting threshold of a predefined value is used after which that particular track terminates and the end time is stored. The algorithm continues to find other tracks relevant to the query.
- Now that we have the start and end timings of all the relevant tracks, merging of consecutive tracks is done if they are supposed to be continuous, based on predefined threshold.
- Finally the retrieved video segments are shown on the original video, so the the user can either choose to continue or could navigate to before or after the specified video segment.



Design Approach

Benefits of the approaches used:

Video Approach:

- Most of the video data models built so far are usually **content and rule based**, wherein objects are detected using bounding rectangles and are primarily used for querying event based and spatial relations. These are usually application specific and domain dependent, and which require certain level of annotation.
- The model prescribed by us does not need annotation of the video database. Instead image captioning tools are employed to extract information about the objects and events through deep neural networks. This information is used for querying which makes our model **domain-independent** and serves general purpose.

Audio Approach:

- This can be used on all videos with audio to seek to the part where anything related to the given query was present (Semantically similar). Eg: College lecture video.



Design Approach

Drawbacks:

1. Neuraltalk2

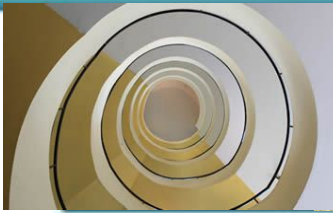
There is lesser localization as the whole track is formed on the basis of a caption. The spatial relations between the objects in the frames are not given importance.

2. Densecap

Tracklets formed can be extremely small.

3. Audio Approach

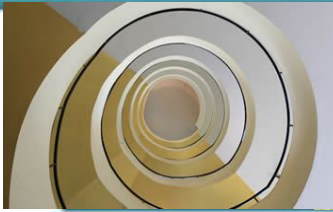
As semantic similarity is the base for forming tracklets, results might not be accurate as other factors in the time domain and frequency domain also need to be considered.



Design Approach

Other efforts in Image Captioning(not employed):

1. We built our own Image captioning model, training it on the Flickr8k dataset, employing a CNN-RNN model.
2. Our Model comprises of two basic components:
 - Convolutional Neural Network: This is used for extracting the most important features of the input image, breaking down an image of size $224*224*3$ into a vector of $1*4096$ recognizable features.
 - Recurrent Neural Network: Input to this component of the model are the features, in which we implement LSTM cells, an improvised concept of RNN involving memory. Input to the cell at time step t will be the caption predicted at time step $t-1$ and the weights are accordingly adjusted to produce the most probable word keeping the embeddings into consideration.

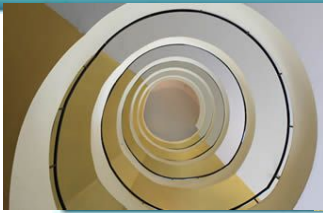


Design Approach

Other efforts in Image Captioning(not employed):

Algorithm -

1. From the available captions, vocabulary is formed by setting a threshold on the frequency of the words, hence forming a vocabulary of 996 words.
2. Each word is mapped into a 256 dimension vector, to represent words with a bigger space.
3. The input image is mapped into a word space to provide appropriate input to the LSTM cell, which forms the input for the first time step.
4. As the loop progresses, to predict the next word, the embeddings of the previous word are passed to the LSTM.
5. Finally, at the end of every epoch, we calculate form a mask vector taking care of how many words are used in the caption and also form a one-hot encoding for the words.
6. Then these features are encoded back into words, Caption is generated using a naive greedy approach.



Design Approach

Other efforts in Image Captioning(not employed):

Results of this approach -

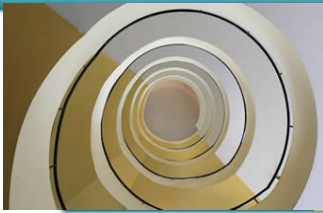


A brown dog runs through a grassy field.
the .



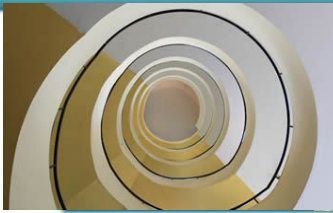
A boy in swimming trunks walking along

But since this model wasn't performing that well, we decided to go ahead with pre-built image captioning models such as Densecap and NeuralTalk2.



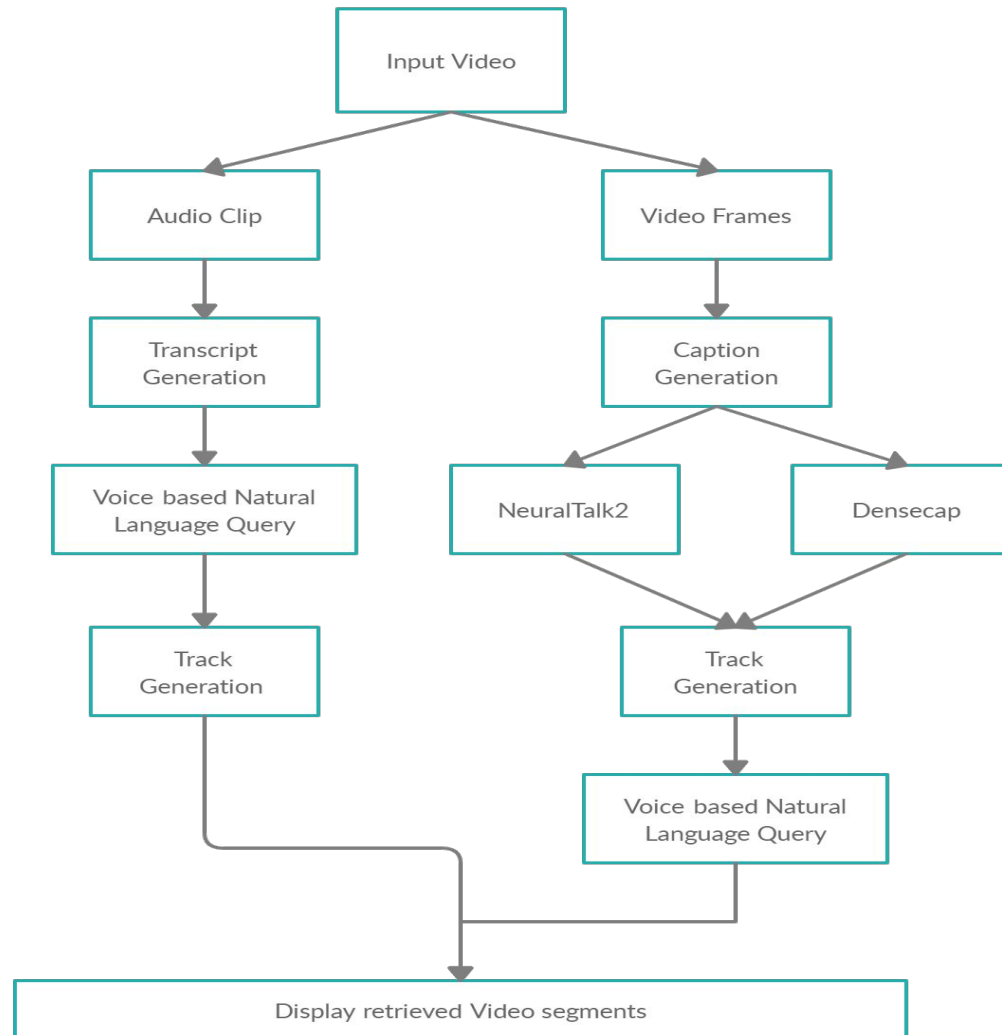
Any obstacles/challenges

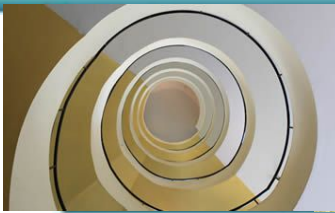
1. The application of video retrieval requires a lot of computational power. **NVIDIA CUDA** enabled GPU was used for better computational abilities. GPU enabled DenseCap and NeuralTalk2 implemented in Torch are used to generate the captioning of the frames.
2. The program to create tracklets was run on **Google Colab**. However just running 10 frames took about 4 hours. And the video we had had 1950 frames. We found a hack to enable 35gb ram TPU. We then decided to divide the entire task into smaller tasks to get intermediate tracks and then later merge them. However, it was still not very feasible with the resources we had.
3. Colab is ineffective as it continuously keeps disconnecting, and since the final requirement is to process more than a single video, colab would be very inefficient. Hence current model is run on a 1 minute video for demo.



Design Description/UI Design

Model:

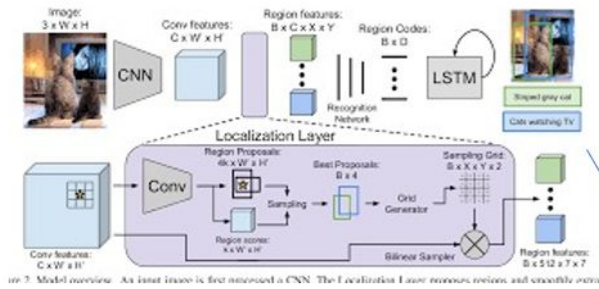




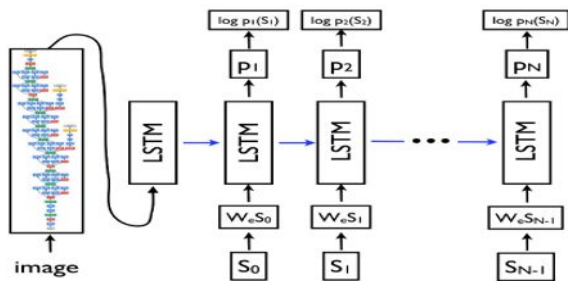
Design Description/UI Design

Video Approach Model:

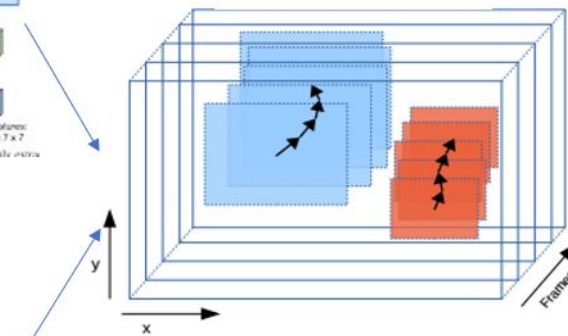
DENSECAP



NEURALTALK2

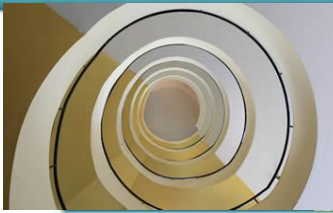


CREATE TRACKLETS



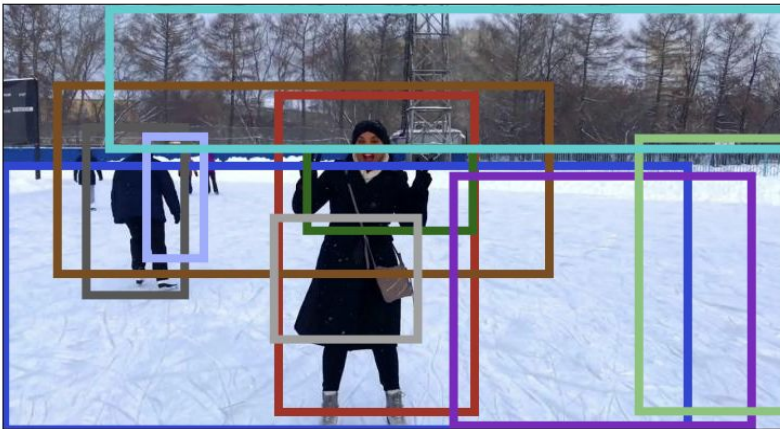
VOICE BASED QUERY





Design Description/UI Design

Image Captioning Model Results:



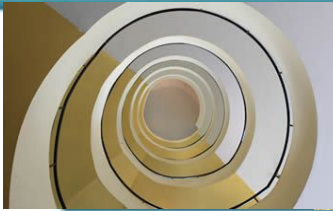
woman wearing black pants. snow on the ground.
person wearing a black jacket. the jacket is black.
people skiing on a mountain. tracks in the snow.
woman is wearing a black coat. snow on the
ground. person wearing a blue jacket. trees with
no leaves.

Captions generated by Densecap



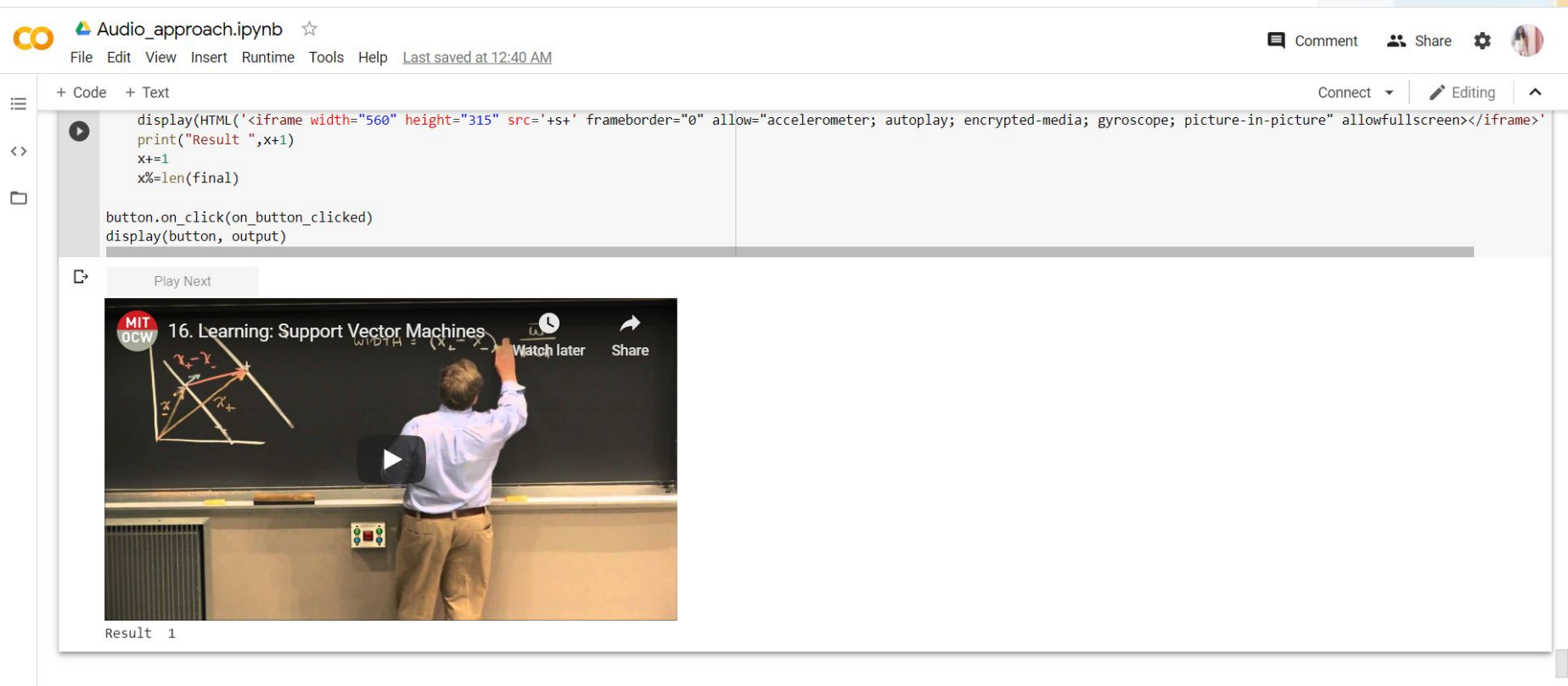
a woman is standing in the snow on skis

Captions generated by NeuralTalk2



Design Description/UI Design

Interface Screenshots:



The screenshot displays a Jupyter Notebook interface. At the top, the title bar shows "Audio_approach.ipynb" with a star icon. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help. The main area is divided into two sections: a code editor and a preview area. The code editor contains the following Python code:

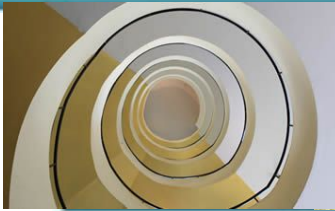
```
display(HTML('<iframe width="560" height="315" src='+s+' frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>'))
print("Result ",x+1)
x+=1
x%=len(final)

button.on_click(on_button_clicked)
display(button, output)
```

The preview area shows a video player with the title "16. Learning: Support Vector Machines" and a play button. The video player also includes a "Watch later" button and a "Share" button. The video player is currently showing a frame of a person writing on a chalkboard.

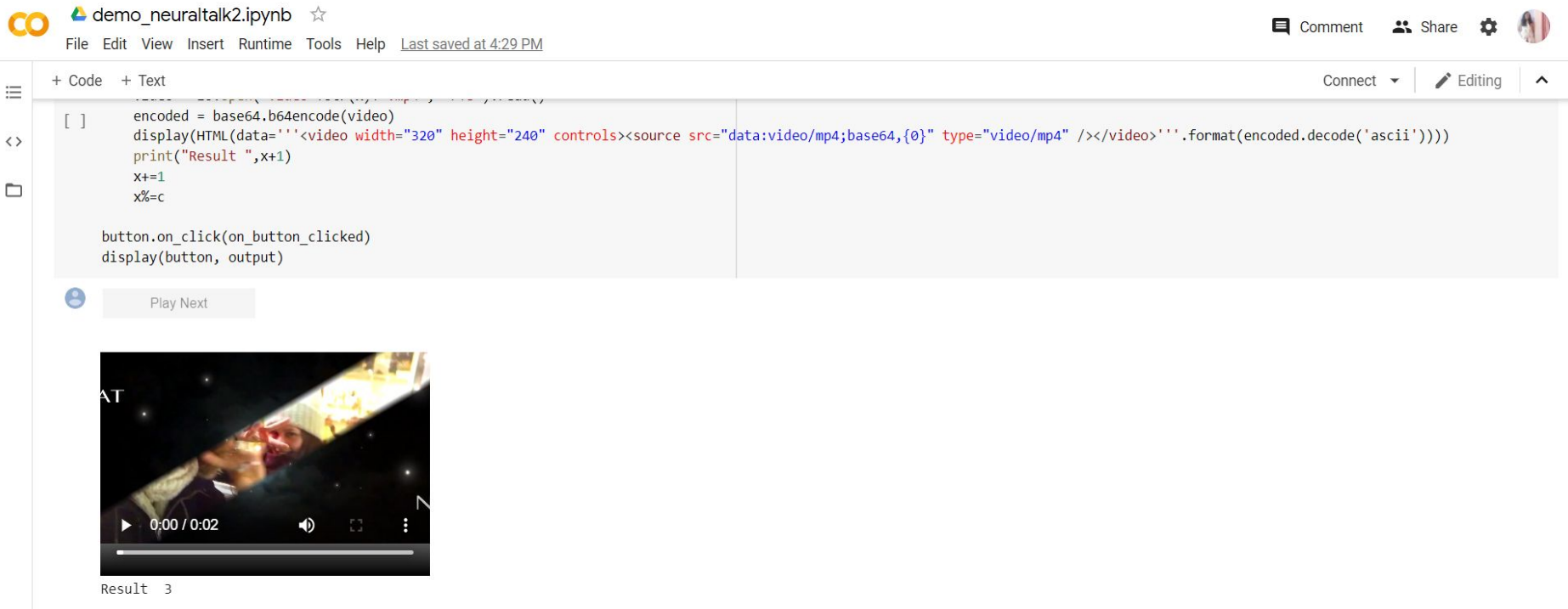
Result 1

Audio Approach Results



Design Description/UI Design

Interface Screenshots:



The screenshot displays a Jupyter Notebook interface. At the top, the title bar shows "demo_neuraltalk2.ipynb" with a star icon. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, and Help. A status bar at the bottom of the menu indicates "Last saved at 4:29 PM".

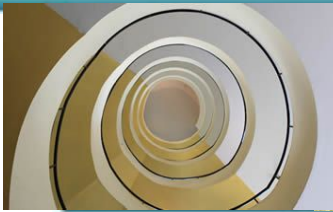
The main area of the notebook is divided into two sections. The top section is labeled "+ Code" and contains the following Python code:

```
[ ] encoded = base64.b64encode(video)
display(HTML(data='''<video width="320" height="240" controls><source src="data:video/mp4;base64,{0}" type="video/mp4" /></video>'''.format(encoded.decode('ascii'))))
print("Result ",x+1)
x+=1
X%=c

button.on_click(on_button_clicked)
display(button, output)
```

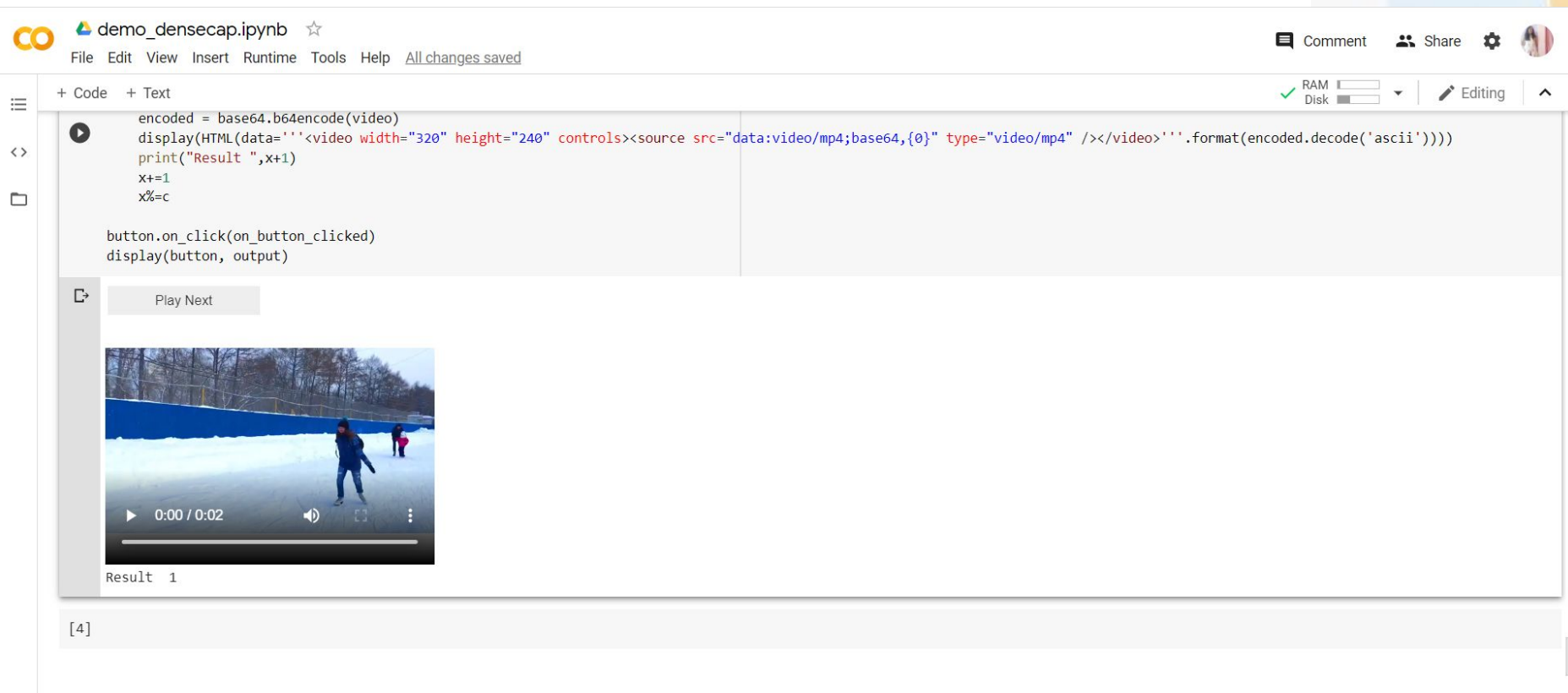
The bottom section is labeled "+ Text" and contains a "Play Next" button. Below the button is a video player showing a scene with people. The video player has a progress bar at the bottom indicating "0:00 / 0:02". Below the video player, the text "Result 3" is displayed.

NeuralTalk2 Results



Design Description/UI Design

Interface Screenshots:



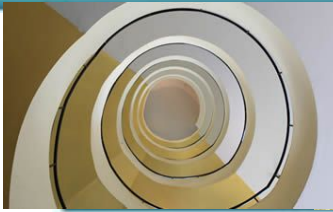
The screenshot displays a Jupyter Notebook interface. The top bar shows the file name "demo_densecap.ipynb" and a star icon. Below the bar are tabs for "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a link to "All changes saved". The main area is divided into two sections: "Code" and "Text". The "Code" section contains the following Python code:

```
encoded = base64.b64encode(video)
display(HTML(data='<video width="320" height="240" controls><source src="data:video/mp4;base64,{0}" type="video/mp4" /></video>'.format(encoded.decode('ascii'))))
print("Result ", x+1)
x+=1
x%=c

button.on_click(on_button_clicked)
display(button, output)
```

The "Text" section shows the output of the code, which is a video player. The video player has a "Play Next" button and a video of a person ice skating. The video player shows a progress bar at 0:00 / 0:02. Below the video player, the text "Result 1" is displayed. At the bottom of the notebook, there is a cell labeled "[4]".

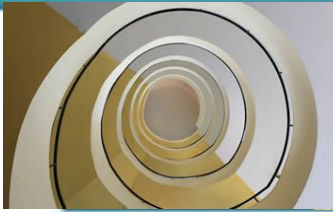
Densecap Results



Project Demo

Video Approach:

- Tested on a one minute video. The number of frames generated was 519 (using opencv).
- The similarity threshold in constructing semantic tracklets varied from 0.6 to 0.8.
- Time taken to find similarity between two sentences was approximately 12 seconds. The cutting threshold was set to 5 frames, and the minimum track size was also set to 5 frames.
- A set of tracks with representative caption was proposed.
- Retrieval Rate : around 17 minutes (NeuralTalk2)
around 3 hours (Densecap)
- Padding was added to the retrieved videos if less than one second.



NeuralTalk2 Approach

Video: <https://youtu.be/rnQLGLvLSIQ>

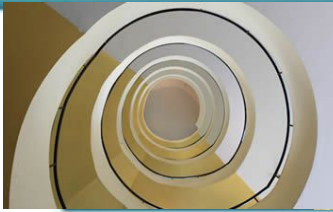
Voice query: “a woman is skiing”

Outputs:

<https://youtu.be/HuUamlDneSg>

<https://youtu.be/HigfoDnZsbY>

<https://youtu.be/t-U2jXjfzik>



Densecap Approach

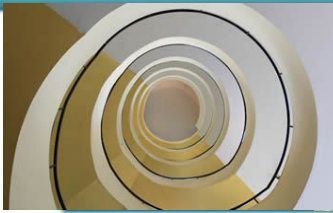
Video: <https://youtu.be/rnQLGLvLSIQ>

Voice query: “a woman is skiing”

Outputs:

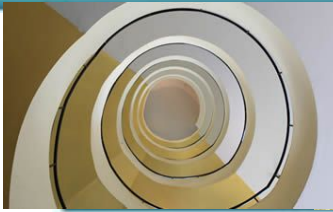
<https://youtu.be/g14sERx98qw>

<https://youtu.be/lq9eHwS6Uq4>



Audio Approach:

- Tested on a 49 minute youtube lecture video on SVM by Patrick Winston. The number of transcripts generated was 868.
- The similarity threshold in constructing semantic tracklets was set to 0.6.
- Time taken to find similarity between two sentences was approximately 12 seconds. The cutting threshold was set to 2 transcripts, and the minimum track size was also set to 7 seconds.
- Different set of tracks relevant to the input query was proposed.
- Retrieval Rate : One day



Audio Approach

Video: <https://youtu.be/PwhiWxHK8o>

(Support Vector Machines by Instructor : Patrick Winston)

Voice query: “constraint”

Outputs:

<https://youtu.be/PwhiWxHK8o?start=462&end=548>

<https://youtu.be/PwhiWxHK8o?start=831&end=876>

<https://youtu.be/PwhiWxHK8o?start=1045&end=1055>

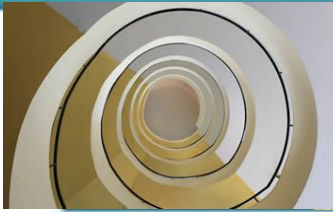
<https://youtu.be/PwhiWxHK8o?start=1134&end=1149>

<https://youtu.be/PwhiWxHK8o?start=1355&end=1604>

<https://youtu.be/PwhiWxHK8o?start=1849&end=1881>

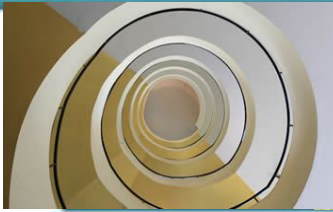
<https://youtu.be/PwhiWxHK8o?start=2058&end=2115>

<https://youtu.be/PwhiWxHK8o?start=2313&end=2380>



Future Scope

1. **Caching (or storing in a DB):**
 - The retrieval rate in Audio approach for a 49 minutes video is around a day for a given query. These results can be cached. **Caching** can drastically increase the performance for repeated queries.
 - Since this approach is used mainly for lecture videos, the results for common keywords can be cached. This would make retrieval rate for them way faster.
2. A web portal can be developed for displaying the results. Right now the results are shown on CLI.
3. The demo is right now showed for particular videos. This can be extended to a complete database.



Summary

1. For the given problem statement we present two broad approaches, Audio and Video approach to retrieve video segments for a given video.
2. The Video approach is performed using two image captioning techniques namely being Densecap and NeuralTalk2. This approach mainly had 3 different phases:
 - i. Generate Captions
 - ii. Creating Tracklets
 - iii. Voice based Search.
3. Audio approach mainly consists of Transcript generation, voice based query and Track generation on fly and results displayed after merge and filter
4. The model performs appreciably well on the inputs provided and tests performed.