Multimodal Analysis for Tagging Coronavirus News Videos From India

Tom Joshi
ttj2108@columbia.edu

Directed Research Report - Spring 2020
Adviser: John Kender
Department of Computer Science
Columbia University

Abstract

With the growth of video archives produced across the world, there is a need to analyze these videos across multiple modalities such as the video's visual and lingual components. We construct an approach focused on news videos, which are rich in both visual and lingual information. The goal is to extract features from videos using computer vision and natural language processing techniques to then use forms of Canonical Correlation Analysis to detect specific tags for news videos produced by different cultural affinity groups. In order to do so, we build the data collection, analysis, and feature extraction pipeline needed to create culture specific tags for videos. An API (Application Programming Interface) and Three Tier Architecture are developed for integration with our web application and for other developers to access our analysis. The analysis we conduct is on news videos on Coronavirus produced by news outlets in India and we test our Web Crawler on the topic of Modi's reelection.

Keywords: Multimodal embedding, machine learning, canonical correlation analysis, news video analysis, news video tagging, computer vision, natural language processing

1 Introduction

Two of the most important applications of machine learning today are in computer vision and natural language processing. Much of computer vision research is conducted on datasets of large amounts of images. Natural language processing tasks are typically conducted on large datasets of text corpuses. However, humans largely interact with their environment in multimodal ways: we synthesize visual and lingual information to develop an understanding of a situation. There has been growing interest in the computer vision and natural language processing communities to develop systems to process many types of information similar to how a human processes across many modalities. Multimodal machine learning aims to build models that accept inputs from multiple modalities (visual, lingual, etc.) and compute relationships between data from different modalities [1]. An example of a combination of lingual features and visual features is in video summary or video tagging. By developing a system to process visual and lingual features we can extract more insight from certain datasets.

News videos have both a visual component (e.g. a clip of a news event) and a lingual component (e.g. broadcasters discussing the importance of such news event). There is a large expansion in the number of news videos produced all around the world. Many of the news videos produced cover similar topics that are pertinent across many cultural affinity groups. The process of analyzing all of this news media manually can be very time consuming and personal biases can also affect interpretations.

In this work, we resolve to automate the process of analyzing news media. We build upon the work done in Tsai [2] and Liu [3]. Tsai introduces a new task of detecting culture-specific tags for news videos. They break down videos into keyframes which are images, create image-text pairs with keyframes and textual descriptions of the original video, and then detect culture specific tags for news videos. In order to do so, they devised a way to measure the semantic similarity between visual and text data. A three-view embedding is developed where images shared in cultures X and Y are paired with their corresponding text description from culture X and Y to form a triplet: (text description from culture X, image from culture X and Y, text description from culture Y). In addition, two-view pair-pair embeddings are used where an image-text pair for two different cultures are analyzed. A joint embedding space is learned as a shared latent space for visual and text features. This shared latent space can be used to compare vectors from different modalities and is created using Canonical Correlation Analysis (CCA). In CCA, we are trying to find two sets of basis vectors, one for some a and for some b. If we are to consider the linear combinations of $a^T\hat{w}_a$ and $b^T\hat{w}_b$, then we attempt to maximize the correlation by maximizing the following function:

$$\rho = \frac{w_a^T C_{ab} w_b}{\sqrt{w_a^T C_{aa} w_a w_b^T C_{bb} w_b}}$$

CCA is a way of finding two sets of basis vectors for two variables such that when the variables are projected to the new space their correlation is optimized. The drawback with CCA is that it is restricted to measuring linear projections, which is addressed in Deep Canonical Correlation Analysis (DCCA). DCCA is the application of deep neural networks to learn nonlinear representations. The two inputs are passed through a neural network of nonlinear transformations to produce a projection. DCCA learns parameters that maximize the correlation between the vectors in their projected spaces. However, DCCA is restricted in the number of inputs variables, whereas Generalized Canonical Correlation (GCCA) is not restricted in the number of variables it takes. GCCA learns a single projection that is independent of any single input variable. CCA, DCCA, and GCCA are used in Tsai's work. Tsai distinguishes their work from previous work by applying tag retrieval to a cultural setting where they analyze different cultural reactions to news events.

Liu's [2] work seeks to extend Tsai's work. Given an image-text pair, they return the most suitable text tag from another culture to describe the image sequence. They attempt to create an automatic pipeline to target and crawl news videos from web archives for different countries. They also constructed a pre-filtering approach to remove irrelevant content from news video for improved data quality. In contrast to Tsai, Liu makes use of the temporal aspect of the speech and images and implements native language embeddings to remove the error that comes with machine translation. Therefore, they update the lingual and visual embeddings models implemented in Tsai's work.

In this work we implement a visual and lingual analysis that feeds into an openly accessible API. We implement a working version of the automatic crawler initiated previously. We build the google cloud architecture required by Liu's work to create transcripts of news videos for the language models. In previous work, the analysis pipeline was disconnected from a web application that displayed results; results had to be manually loaded from the analysis to the web application. We design and build a RESTful API [5] which is the predominant model for accessing and transporting data over the web. The results of our analysis can be accessed by anyone via a call to our API. We extend the RESTful API to construct a Three-Tier Architecture [6], which serves as the client-server software architecture that pervades the modern web. Using the architecture, a connection between our analysis and web application can be made so the web application can at any time make a request to the analysis pipeline and serve results to a user. In order to test our pipeline, we conduct an analysis of the two main news sources in India and their response to Coronavirus and also test our web crawler on the news sources responses to the reelection of Modi.

2 Related Works

Vasili et al. develop a multimodal framework to generate coherent descriptions of online videos from the dataset MSR-VTT [16]. Their work builds on the S2VT model, which implements an LSTM (Long Short-Term Memory) encoder-decoder architecture [11]. LSTM is a specific type of Recurrent Neural Network. Recurrent Neural Networks take each input word as a word embedding and a hidden state at each time step in the input sequence. The word embedding is a vector representation of the word often created by a simple neural network as is done in Word2Vec [14]. LSTM improves upon Recurrent Neural Network by incorporating information about long term dependencies. For example, if in order to predict the last word of a sentence the first word needs to be incorporated, the Recurrent Neural Network model would not be able to effectively incorporate the first word to predict the last word because of its issue with vanishing gradients [15]. LSTM addresses this issue by maintaining the old states throughout the neural network. The image frame is encoded by the LSTM into a fixed dimensional vector and then descriptions are produced word by word from the decoder. During the encoding phase, an LSTM model creates a sequence of hidden states from the input image frames. Then, the decoder produces words given information from the hidden states and the previously produced word. Vasili et al. extend the approach by inputting not just visual data but multimodal data. They utilize object recognition features from the state-of-the-art Resnet, action recognition features proposed for C3D [12], video category information, and audio features derived from Mel Frequency Cepstral Coefficients.

Note that our work differs from this work in a few ways. We seek to generate specific tags for news videos from different cultural affinity groups whereas Vasili et al. [16] seek to generate sentence descriptions for a wide range of videos. Second, they combine the features simply by concatenating the features together. However, we implement a multimodal representation by using a joint representation through our use of various forms of Canonical Correlation Analysis [1].

Jin et al. [13] also create a model to create video descriptions using a LSTM network. MSR Video To Language Challenge dataset is used and the features inputted into the model are image modality features from VGG-19 trained on Imagenet, video modality features from C3D, aural modality features, speech modality features from IBM Watson API's transcription of each video, and a meta modality feature such as the category of video.

Their work differs from our work in their multimodal representation. They use a fusion network as an encoder that takes in their multimodal inputs and produces a fixed length output vector. During training, they learn parameters of the encoder decoder structure. This structure once again is in contrast with our joint representation.

3 News Event Selection

In order to test our cultural affinity groups investigation, we evaluate topics that have two cultural affinity groups that hold differing opinions on a highly covered news topic. After manual investigation of news topics from the past few years, we decide that Coronavirus and the Reelection of Modi would be suitable topics. The topics have divided cultural affinity groups that could be represented through two prominent news organizations in India, NDTV and Republic World. Both organizations have copious amounts of news videos published to YouTube in English.

4 Methods

4.1 Data Collection

Crawler We are able to restrict our download source to YouTube. We first extract video metadata from YouTube's API. We enable our Spider class to specify a specific YouTube channel and our get requests to the API contain the channel IDs of NDTV and Republic World which are extracted from their channel page source code. In order to prevent YouTube from providing videos from extraneous sources. A channel specific approach improves upon the noise prone method of collection process in [3] which uses the result of Most Relevant Videos filtered to include Medium-length videos. We specifically query for "NDTV Coronavirus" and "Republic World Coronavirus" for example. Once the API responds, we use the results to build a URL for each video, which is then passed to a third-party library to extract the videos from the URL. Using the third-party library, we extract the visual and audio component of the video separately to initiate the two coinciding pipelines for visual and lingual feature extraction.

4.2 Language Preprocessing

Transcript Creation The audio files are extracted from the video using ffmpeg library. FLAC is used as the audio format because it is a lossless codec as opposed to mp4 for example. Speech recognition can be a computationally intensive task and words can be undiscernible with too much noise. Therefore, Google Cloud's Speech API program is implemented as it can handle slightly noisy audio data and perform computation quickly. Because our NDTV and Republic World videos were in English we did not need to support multiple languages. As in standard natural language processing studies, words are tokenized and stemming and lemmatization are applied.

4.3 Visual Preprocessing

Pairing Images with Text As in Liu's work [3], the chronological order of the images and texts are maintained. Text from the transcript of a video is paired with a corresponding image from the video based on the timestamp in the audio transcript. Every five seconds, we sample n frames from the video file to be paired with the corresponding text in the transcript. Frames are only dropped if there is no text for a particular image. In order to identify differences in reactions to similar events, we identify near duplicate keyframe pairs between the NDTV results and Republic World results as in [3] with greedy search. To measure the similarity between image frames, the visual feature vectors were compared using the cosine similarity as opposed to spatial distance for its robustness against variance and frequency.

4.4 API

In Liu's work [3], the analysis is not connected to the website that was built to show results. The results must be manually moved from the analysis to the website. In addition, there may be other developers who would like to take advantage of the analysis pipeline. A RESTful API would provide an interface for the research project's website and external developers. RESTful APIs' are the predominant form for web-based communication between programs on the internet today. A user could provide a research topic such as "Coronavirus NDTV" and "Coronavirus Republic World".

In return the user would receive JSON formatted in three main parts: the image-text pairs of the first cultural affinity group, the image-text pairs of the second cultural affinity group, and the common images with their respective text. Each image text-pair should return the results of time of the video released, images, text corresponding to the images, and information for the hyperlinks to the videos. There are several API design principles that we adhere to.

(1) Request URLs should include resources pluralized to specify that the user is accessing a collection of resources. For example, "GET /videos" would be preferred over "GET /getvideo/".

(2) Part of the path should not be used to specify resources. Instead query parameters should be used to specify resources, which allows the user to filter by multiple filters at a time. For example, "GET /videos?query1=Coronavirus+NDTV" would be preferred over "GET /videos/Coronavirus+NDTV".

(3) Paths should remain flexible to future changes such as if more resources or functionality are added to the API.

(4) Paths should contain version numbers so that programs built on older version of the API do not stop working from a change in the underlying implementation. For example, "GET /v1/resources/videos?query1=Coronavirus+NDTV" would be an example of a properly structured API.

With these design principles in mind, a RESTful API is built to access data produced in this study. There are seven methods that can be used as gateways for any developer or website (such as our group's website) to retrieve information. The seven methods are all GET methods built according to the information listed as necessary to build the website in Jiaqi Liu's work [17]. The seven possible access points are:

GET /v1/resources/cross-cultural-analysis/image-text-pairs

- Takes the arguments topic. Returns image-text pairs, the timestamp of the similar frame, and the similarity of the frames.

GET /v1/resources/cross-cultural-analysis/processed_transcript

- Takes the arguments topic and group (cultural affinity group). Returns the processed transcripts of videos with timestamps for each word.

GET /v1/resources/cross-cultural-analysis/raw_transcripts

- Takes the arguments topic and group. Returns Google Speech-To-Text API results

GET /v1/resources/cross-cultural-analysis/frames/all

- Takes the arguments topic and group. Returns a list of names for all the frames from the videos

GET /v1/resources/cross-cultural-analysis/frames

- Takes the arguments topic, group, and filename. Returns specific video frame.

GET /v1/resources/cross-cultural-analysis/videos˙info

- Takes the arguments topic and group. Returns the id of video needed to create URL and video title

The methods are included in Figure 9, 10, 11 as documentation for usage.

4.5 Three Tier Architecture

With an API developed, the cultural analysis pipeline could be directly connected with the backend of the web application we developed. In order to do so effectively the Three Tier Architecture should be implemented. The Architecture includes three components: the Presentation Tier, the Logic Tier, and the Data Tier as depicted in Figure 1. The Presentation Tier is the tier that the user interacts with. This displays information such as those discussed in our group's previous work developing a web site. The information includes similar images used in differing cultural affinity groups and texts that coincide with images. However, I will not be going into depth about this layer in this paper. The Application Tier controls the logic of the programs such as making the proper requests to the API. The Data Tier is the tier that is detailed in this paper. It handles the storage and data creation mechanisms and should provide an API that the Application Tier can interact with. The Application Tier can use these methods to store data without creating dependencies on any particular mechanism used in the Data Tier. The Data Tier should not have to make POST requests to the Application Tier, such as a Flask-based web app, as the Data Tier should be open to working with many different programs rather than being built for a single program.

5 Experiments

The Youtube API was invoked to return results for Coronavirus from NDTV's Youtube channel and Republic World's Youtube channel. The videos were then downloaded and the audio was extracted using ffmpeg. With the audios in FLAC audio format, the transcripts were sent to and stored in a Google Cloud bucket to be processed by Google's Speech-to-Text API. The API returned JSON formatted transcriptions to the analysis pipeline. The transcripts return had two parts. The first part was the words spoken in the audio. The second part was the timestamp that each word started, ended, and the confidence the API had in each word. The two parts are shown in Figure 2 and 3.

The transcripts are then processed by taking the average of the end and beginning of the word and converted from milliseconds to seconds. A json file is produced for each processed transcript that includes the transcripts name, each word, and the average timestamp of each word. The frames are then extracted at particular timestamp specified by the processed transcripts. The frames are then paired up with the text at the specified timestamp from the processed transcript as shown in Figure 4.

Visual features are then extracted from the frames using VGG-19 model. The duplicate frames are found by angular similarity. A JSON file is produced in the format: [name of the video in the first cultural affinity group, timestamp of similar frame, name of the video in the second cultural affinity group, timestamp of similar frame, similarity of the two frames] The JSON produced is rendered in Figure 5. The results of the API endpoints are shown in Figures 6, 7, and 8.

6 Conclusion

The goal is to build a pipeline for multimodal machine learning analysis. Multimodal analysis is conceived as a way of performing tagging on cross cultural new videos. In order to perform tagging, news videos must be collected, processed, and features must be extracted. In this work, the data processing, analysis, feature extraction, and image-text pair similarity results became operational. An API is built which enables the construction of a Three Tier architecture paired with the web application that we have built. For future work, there should be a pipeline between the feature extraction results and the application of models such as BERT and Canonical Correlation Analysis. One should be able to run each of these steps in one process. Second, the channel ID is currently retrieved manually from the channel page on YouTube. The user should be able to specify a specific YouTube channel and the channel ID should be accessed automatically from YouTube's API. Third, the web application should be changed to take advantage of the API developed for accessing this studies data.

7 References

[1] T. Baltrusaitis, C. Ahuja, and L. Morency. 2017. Multimodal Machine Learning: A Survey and Taxonomy.

[2] C. Tsai, J. Kender. 2017. Detecting Culture-specific Tags for News Videos through Multimodal Embedding.

[3] Y. Liu. 2019. Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups.

[4] G. Andrew, R. Arora, J. Bilmes, and K. Livescu. 2013. Deep Canonical Correlation Analysis. ICML.

[5] R. Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures.

[4] W. Eckerson. 1995. Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications.

[9] M. Borga. 2001. Canonical Correlation a Tutorial.

[10] J. Xu, T. Mei, T. Yao, and Y. Rui. 2016. MSR-VTT: A Large Video Description Dataset for Bridging Video and Language. IEEE Conference on Computer Vision and Pattern Recognition.

[11] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko. 2015. Sequence to Sequence – Video to Text.

[12] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In IEEE International Conference on Computer Vision.

[13] Q. Jin, J. Chen, S. Chen, Y. Xiong, A. Hauptman. Describing Videos using Multi-modal Fusion.

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality.

[15] R. Jozefowicz, W. Zaremba, I. Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures.

[16] V. Ramanishka. A. Das, D. Park, S. Venugopalan, L. Hendricks, M. Rohrbach, K. Saenko. 2016. Multimodal Video Description.

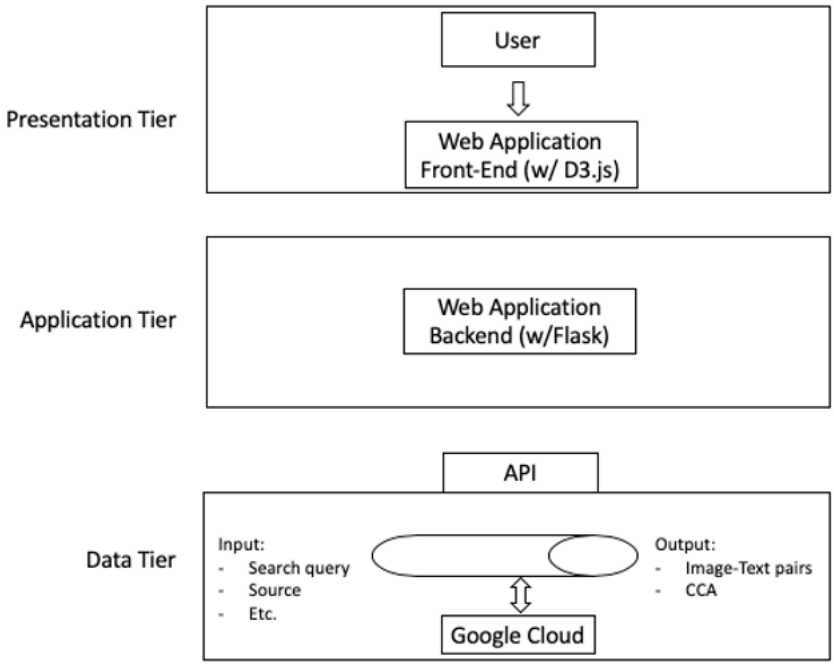

Figure 1: Depiction of Three Tier Architecture implemented in this project


```
[{"transcript": "unwell on Republic TV have been taking your questions in lockdown top experts who are
```


Figure 2: Example of raw transcript


```
"words": [{"word": "unwell", "start": 0.0, "end": 0.8, "confidence": 0.9876290559768677}
```


Figure 3: Example of processed transcript

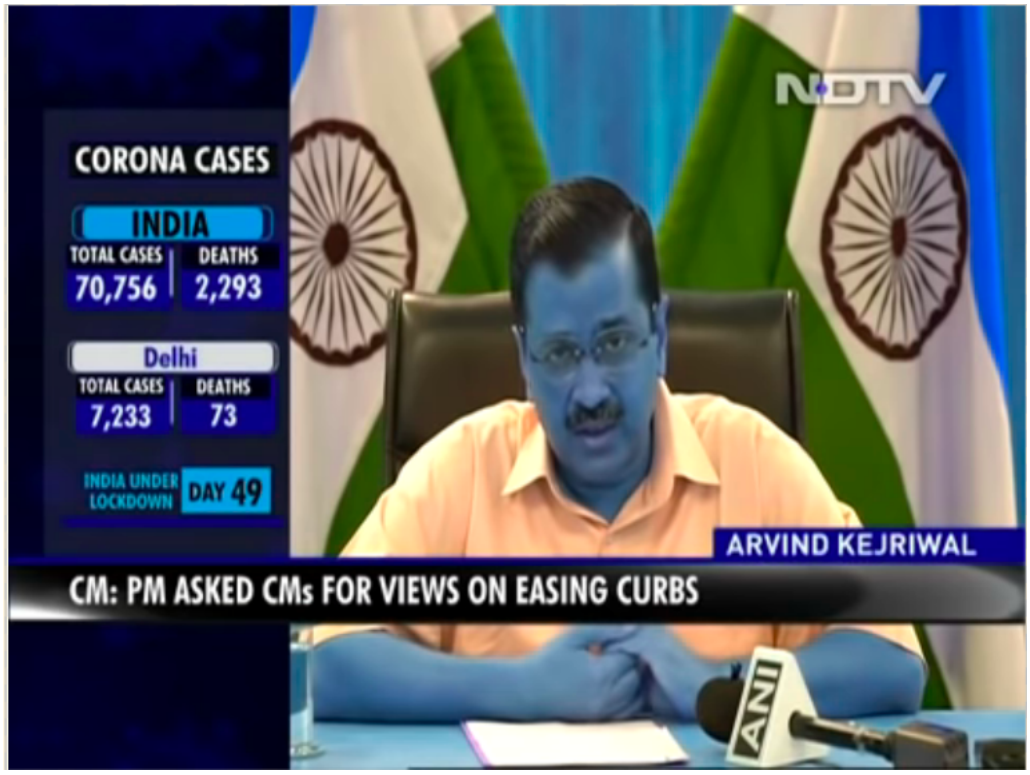

Figure 4: Example of image extracted from video

results > Coronavirus > {} pairs.json > ...

```
1   [["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 9750, 0.056579423011924646], ["xNsYJDwW4Ec", 290200,
    "tE3syQaf10g", 10200, 0.05659283200135772], ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 900, 0.
    05663948124475152], ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 7250, 0.056765076091302054],
    ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 21600, 0.056783043283368 73], ["xNsYJDwW4Ec", 96100,
    "tE3syQaf10g", 9750, 0.05683818757454273], ["xNsYJDwW4Ec", 96550, "tE3syQaf10g", 9750, 0.
    05693029552141251], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 7250, 0.05696771926623987],
    ["xNsYJDwW4Ec", 96550, "tE3syQaf10g", 7250, 0.05699329453013203], ["xNsYJDwW4Ec", 290200,
    "tE3syQaf10g", 21050, 0.05704983331826158], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 10200, 0.
    05706760603966438], ["xNsYJDwW4Ec", 96550, "tE3syQaf10g", 10200, 0.05711376199161813],
    ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 10450, 0.057126729864667304], ["xNsYJDwW4Ec", 96550,
    "tE3syQaf10g", 22900, 0.05715807284385859], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 24700, 0.
    05716157805899514], ["xNsYJDwW4Ec", 96550, "tE3syQaf10g", 24700, 0.05716221838922983],
    ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 22900, 0.057167739458808924], ["xNsYJDwW4Ec", 290200,
    "tE3syQaf10g", 24700, 0.05720861624371668], ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 12000, 0.
    05721625751785064], ["xNsYJDwW4Ec", 272850, "tE3syQaf10g", 9750, 0.05723047759224765],
    ["xNsYJDwW4Ec", 96550, "tE3syQaf10g", 6850, 0.0572475578083596], ["xNsYJDwW4Ec", 285900,
    "tE3syQaf10g", 9750, 0.05724808430210812], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 22900, 0.
    05727904782634551], ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 6850, 0.0572911334666269],
    ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 16000, 0.05729643160642058], ["xNsYJDwW4Ec", 96550,
    "tE3syQaf10g", 21600, 0.05729939609824784], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 6850, 0.
    057317728515707635], ["xNsYJDwW4Ec", 272850, "tE3syQaf10g", 900, 0.0573539474910564],
    ["xNsYJDwW4Ec", 285900, "tE3syQaf10g", 10200, 0.05736378960392291], ["xNsYJDwW4Ec", 284049,
    "tE3syQaf10g", 9750, 0.05739088268763063], ["xNsYJDwW4Ec", 96100, "tE3syQaf10g", 900, 0.
    05741362626892938], ["xNsYJDwW4Ec", 272850, "tE3syQaf10g", 10200, 0.05741521049336187],
    ["xNsYJDwW4Ec", 290200, "tE3syQaf10g", 11600, 0.05742821631190644], ["xNsYJDwW4Ec", 284049,
    "tE3syQaf10g", 7250, 0.05745464534944485], ["xNsYJDwW4Ec", 272850, "tE3syQaf10g", 7250, 0.
```


Figure 5: Example of image extracted from video

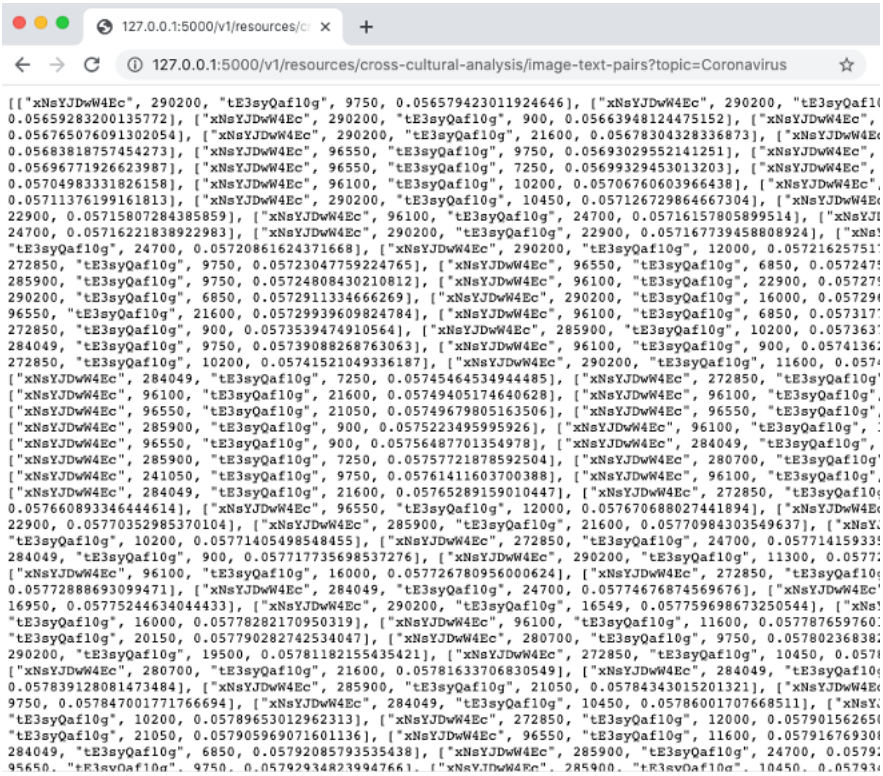

Figure 6: Result from GET /v1/resources/cross-cultural-analysis/image-text-pairs endpoint

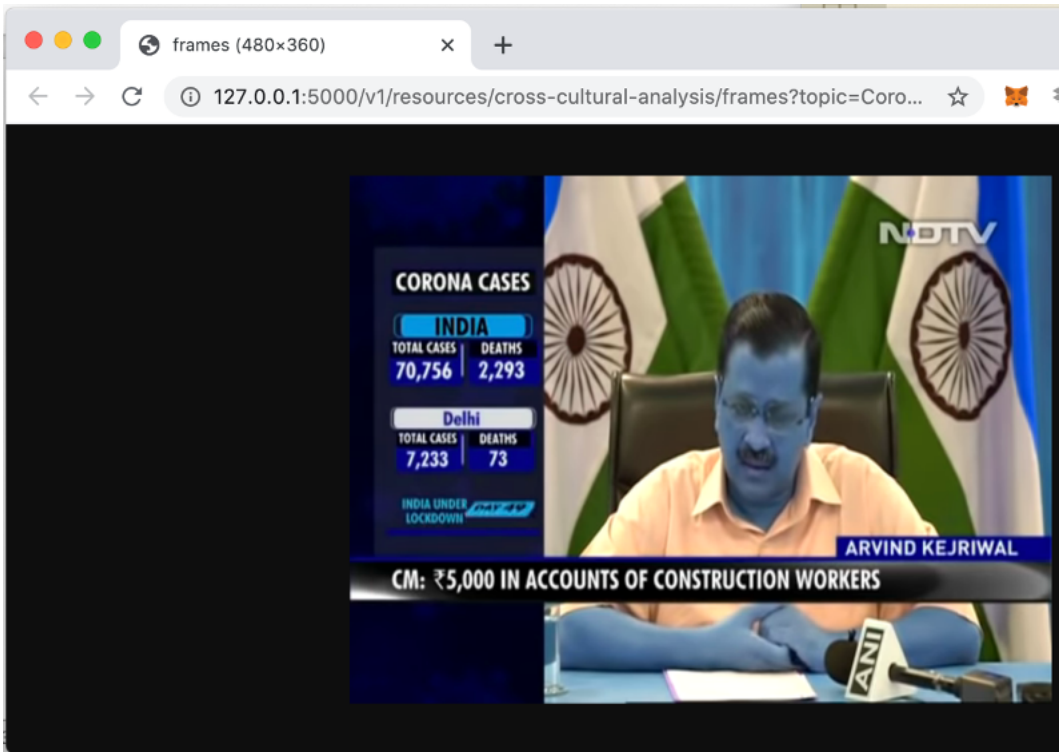

Figure 7: Result from GET /v1/resources/cross-cultural-analysis/frames endpoint

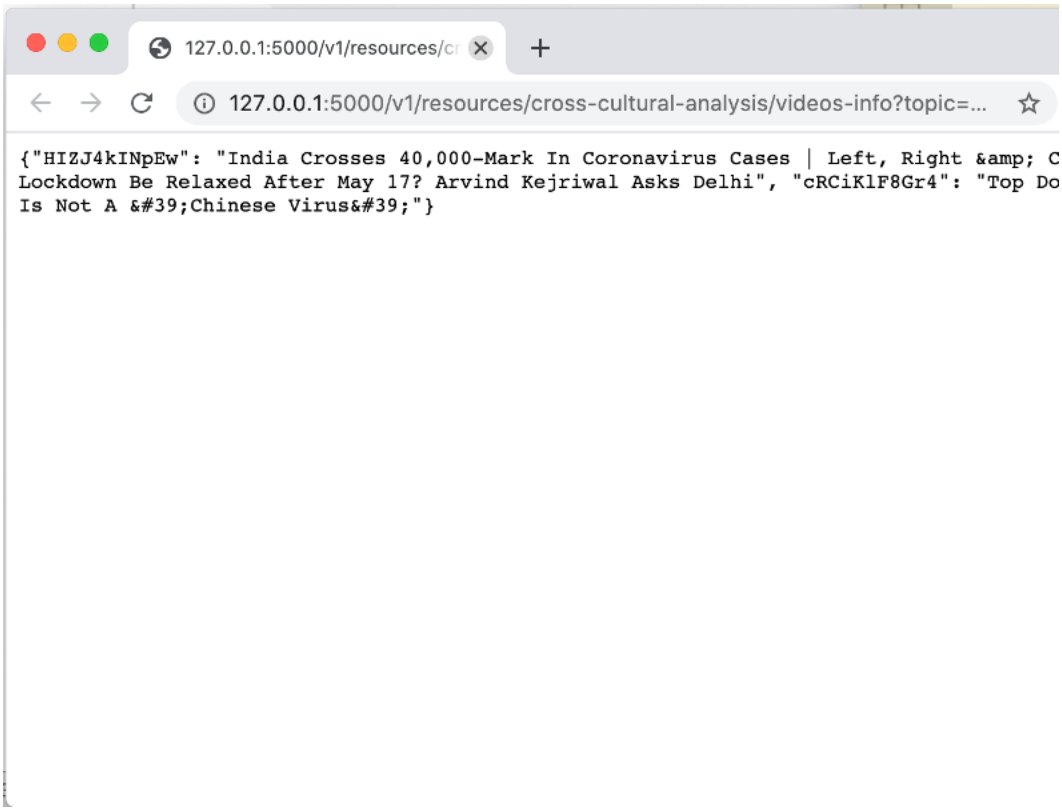

Figure 8: Result from GET /v1/resources/cross-cultural-analysis/videos-info endpoint

```python
# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/image-text-pairs?topic=Coronavirus
@app.route('/v1/resources/cross-cultural-analysis/image-text-pairs', methods=['GET'])
def image_text_pairs():
    if 'topic' in request.args:
        topic = request.args['topic']
    else:
        return "Error: No topic was provided. Please specify a topic"
    # image_text_path = '../results/'+topic+'/pairs.json'
    # with open(image_text_path, 'r') as f:
    #     return send_file(image_text_path, mimetype='application/json')
    image_text_path = '../results/'+topic
    return send_from_directory(image_text_path, 'pairs.json', mimetype='application/json')

# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/processed_transcript?topic=Coronavirus&group=NDTV
@app.route('/v1/resources/cross-cultural-analysis/processed_transcript', methods=['GET'])
def processed_transcript():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
    else:
        return "Error: No topic and/or group was provided. Please specify a topic and group"

    processed_transcript_path = '../results/'+topic+'/'+group
    return send_from_directory(processed_transcript_path, 'processed_transcript.json', mimetype='application/json')

# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/raw_transcripts?topic=Coronavirus&group=NDTV&filename=xNsYJDwW4Ec.json
@app.route('/v1/resources/cross-cultural-analysis/raw_transcripts', methods=['GET'])
def raw_transcript():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
        filename = request.args['filename']
    else:
        return "Error: No topic and/or group and/or filename was provided. Please specify a topic, group, and filename"

    raw_transcript_dir = '../results/'+topic+'/'+group+'/transcripts'
    return send_from_directory(raw_transcript_dir, filename, mimetype='application/json')
```

Figure 9: API Code Part 1

```python
# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/raw_transcripts/all?topic=Coronavirus&group=NDTV
@app.route('/v1/resources/cross-cultural-analysis/raw_transcripts/all', methods=['GET'])
def raw_transcript_list():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
    else:
        return "Error: No topic and/or group was provided. Please specify a topic and group"
    results = []
    raw_transcript_dir = '../results/'+topic+'/'+group+'/transcripts'
    for filename in os.listdir(raw_transcript_dir):
        results.append(filename)
    return str(results)

# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/frames?topic=Coronavirus&group=NDTV&filename=xNsYJDwW4Ec_95650_set.png
@app.route('/v1/resources/cross-cultural-analysis/frames', methods=['GET'])
def frames():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
        filename = request.args['filename']
    else:
        return "Error: No topic and/or group and/or filename was provided. Please specify a topic, group, and filename"

    raw_frames_dir = '../results/'+topic+'/'+group+'/frames'
    return send_from_directory(raw_frames_dir, filename, mimetype='image/png')

# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/frames/all?topic=Coronavirus&group=NDTV
@app.route('/v1/resources/cross-cultural-analysis/frames/all', methods=['GET'])
def frames_all():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
    else:
        return "Error: No topic and/or group was provided. Please specify a topic and group"
    results = []
    raw_frames_dir = '../results/'+topic+'/'+group+'/frames'
    for filename in os.listdir(raw_frames_dir):
        results.append(filename)
    return str(results)
```

Figure 10: API Code Part 2

```python
# http://127.0.0.1:5000/v1/resources/cross-cultural-analysis/videos_info?topic=Coronavirus&group=NDTV
@app.route('/v1/resources/cross-cultural-analysis/videos_info', methods=['GET'])
def videos_info():
    if 'topic' in request.args and 'group' in request.args:
        topic = request.args['topic']
        group = request.args['group']
    else:
        return "Error: No topic and/or group was provided. Please specify a topic and group"

    raw_frames_dir = '../results/'+topic+'/'+group+'/video_metadata'
    return send_from_directory(raw_frames_dir, 'titles.json', mimetype='application/json')


app.run()
```

Figure 11: API Code Part 3