Video Clip Retrieval

Shashi Bhushan Yadav, Oleksandr-Patrik Yandola, Junhao Lin

Northeastern University, Boston, MA
yadav.shas@northeastern.edu, yandola.o@northeastern.edu, lin.junh@northeastern.edu.

SECTION 1

Abstract

In the report, we present and evaluate several video retrieval and ranking algorithms. Our method is based on the metadata and video content and offers annotations for each video. We'll employ supervised and unsupervised techniques for feature vector extraction, and unsupervised algorithms will help with ranking calculations. These solutions were designed to take a text or video query and discover the most comparable videos in our database. Later, we compare these models' performances and combine them to improve one another. Finally, we evaluate the effectiveness of the combined model using precision metrics.

Introduction

There is potential for searching videos for specialized content generation due to the expansion of social media and online video content. Optimizing the results to be current, quick, and pertinent is important so the user can find their desired output in a large database. Ranking video material is one approach to achieve this, but it is difficult because it calls for linkages, web page metadata, and other factors.

We attempt to return the most pertinent videos when given a video or text query as input. Our approach focuses on resolving two key issues. The first approach uses supervised and unsupervised strategies to deal with similar videos regarding appearance and content. Because we do not consider outside variables like weblinks, clickthrough rates, or other sources that could provide information on the user's rankings of the images in that class, we refer to this approach as unsupervised. In this case, the objective is to use metadata and unsupervised machine learning to return the best videos for a particular topic, much as searching for a hashtag's position on Instagram or TikTok.

However, for the time being, we have kept it straightforward by using the videos and their corresponding annotations and metadata. Our system is adaptable to new inputs like synonyms of the categories and their location.

Background

In the feature extraction approach, we used a deep neural network model for encoding videos into feature representations. Our model, in this case, consists of a pretrained network of resnet50 on top of 4 more layers added to it for our use case. The network takes the input image of size (224,224,3) and is 54 layers deep such that the first layer has a 7*7 filter and stride 2, the second layer is 3*3 max pool, stride 2, then there are 3 blocks of 3 layers 2 with 1*1 and filter size 64 and 256 respectively, sandwiching a 3*3 layer with 64 filters. Similarly, there are 3 more blocks for convolutional layers with channel size increasing to 2048; in the end of the Resnet50 model, there is one fc layer of size 1000.

The architecture for the Resnet model is represented in the diagram below.

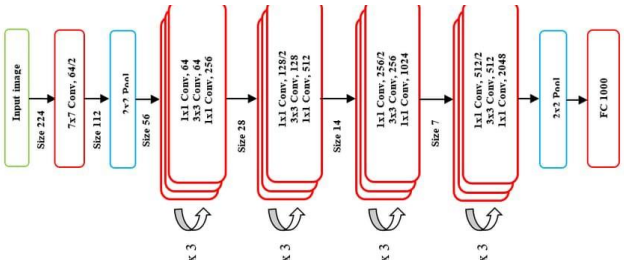


Figure 1: Resnet50 architecture

source:https://miro.medium.com/max/1100/1*rPktw9-nz-dy9CFcddMBdQ.webp

Below is an illustration of cosine similarity, which was used to order the findings from most similar to least similar:

$$cosim(x, y) = \frac{x.y}{||x||||y||}$$

The angle between two vectors is calculated using this evaluation metric. In feature space, more similar or close-together vectors have a greater value. Its value is between 0 and 1.

We utilized Precision, a measure of the percentage of the truly correct outcomes, to evaluate the performance of our algorithms. In other words, the number of items that were found to be relevant out of all the elements is what we may use to define precision:

$$Precision = \frac{TP}{TP+FP} = \frac{Total\ relevant\ items}{Total\ retrieved\ items}$$

Another metric, called precision@k, is a variant of precision for k retrieved items. A helpful metric called Mean Average Precision (mAP) takes the mean of average precision values distributed across the categories:

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$$

Batch gradient descent is used as an optimizer while training. It is denoted by the given formula:

$$\theta = \theta - \eta . \partial_\theta J(\theta; x^{(i)}; y^{(i)}).$$

While the cross entropy is denoted as:-

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i . log(p(y_i)) + (1 - y_i) . log(1 - p(y_i))$$

For the text model part of this project, the Query Likelihood equation was used with Dirichlet smoothing. Log Likelihood itself is based on the Naive Bayes equation.

Bayes' Rule:

$$p(D|Q) \overset{rank}{=} P(Q|D)P(D)$$

This rule states that the probability the document is relevant given a query is the same as the probability a query can be made from the document times the probability of the document itself. Since all the documents are equally likely, the probability of the document is uniform and thus doesn't matter for the sake of comparing documents.

As for the probability of the query given in the document, the equation used is the Query Likelihood. Query Likelihood breaks down the query into separate smaller tokens. Then it adds up the probabilities that all those tokens can be generated by the document (the document, in this case, being the video caption). The formula is:

$$\log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|C|}}{|D| + \mu}$$

Where μ is the average number of tokens per document. Here μ is used as part of Dirichlet smoothing, to avoid zero probabilities for words that don't appear in the document but might appear in the query.

[0,1] Normalization used on the score for combined model is denoted by :-

$$z_i = (x_i - min(x))/(max(x) - min(x))$$

Related work

The methods we use in this project are already widely used in modern video retrieval tasks: keyword-based search and content-based search. The classification methods behind them are also widely used. Most notable is in our ResNet model, where our classification of videos was based on separating frames of a video, converting them into a vector, and calculating the cosine similarity of the input vector to other vectors converted from videos in our dataset.

Project description

The dataset is a set of 7010 short video clips from MSRVTT dataset of varying lengths, 12 seconds on average. These clips include various footage without any specific subject-stock footage, documentary pieces, chunks of advertisements, online videos, clips from shows and movies, etc. These videos have been labeled by humans, and the labels are short sentences with punctuation already removed describing what is happening in the clip in the English language. There are 20 such short descriptions for each video in the dataset. Furthermore, videos have pre-labeled

categories they fit into. There are 20 categories total: music, people, gaming, sports/actions, news/events/politics, education, tv shows, movie/comedy, animation, vehicles/auto, howto, travel, science/technology, animals, kids/family, documentary, food/drink, cooking, beauty/fashion, and advertisements. These categories will be used in the video portion of the project.

The project is divided into 3 sections - the text, video, and combined models. Each of these models corresponds to different queries the user could use. For the text model section, the user can enter a text query and receive the 10 most relevant videos from the dataset of 7010 total videos. For the video model, the dataset consists of only 50 videos, a subset of the 7010 larger dataset. This is due to the restrictions of time and computational power - the model for video comparison takes a long time to train; thus, it was impossible to extend the video model to work on the entire dataset of 7010 videos with the available personal computers in the allotted time for the project. With time and more computing power, however, the video model is scalable to a larger dataset. The text model works on both the larger 7010 videos dataset and the smaller 50 video one that the video model works on. The 50-video batch set was chosen to have videos from 5 classes (music, animation, travel, vehicles, food/drink). The combined text-video model works on the same 50-video dataset on which both text and video work.

For the text-only model, the user enters a query as a string representing what they want to search for, such as “cute cat video,” “birds,” or “driving a car.” For the video-only model, the user sends a query as the name of the video they want to find similar videos of. For example, if the user has a video of a man playing soccer and wants to find similar videos to it in our dataset - they would pass the id of that soccer video to the video model, and the video model would return all the relevant videos they have from the 50-video dataset. For the combined text and video model, the query includes both the text query and the id of a video, where the text query is sent to the text model, and the video query is sent to the video model. The results of the two models are combined, and the user receives the most relevant videos based on their text and video description. In this example, the user can enter the string “white car,” and a video id of a video where someone is driving, and the model will attempt to output a video of a white car being driven.

All three models return the results in a similar way - the top 10 results ranked from 1 to 10. The results are ranked based on relevancy to the query, with the first result being the most relevant and the last being the least relevant. The results are represented as video_ids, and the user can then go to the dataset to search for the returned ids to watch the videos. For all three models, the entire database is ranked for each query. How many top-ranked values are returned can be changed easily in the future by changing a single variable, but the top 10 seemed like a reasonable value for our current dataset.

The text model uses 20 labeled captions for each video to build the model. For each video, the 20 captions are first compressed into a single string, after which the description string is transformed into a frequency dictionary so that each video links to a dictionary of every word in its caption and how often that word is used. A larger collection dictionary is also built, with every unique word used in the captions for all the videos and their counts. Then, each query is compared to every document using the query likelihood method with Dirichlet smoothing. Through the query likelihood method, we assign the score to every document, which represents the sum of the natural log of probabilities that each word in a query can be constructed from the video description language model. The Dirichlet smoothing deals with words that don’t appear in the language model to avoid the zero probability value. Once a dictionary is constructed for the query that maps each video in the database to its query likelihood score, it is sorted in decreasing order. The 10 videos with the highest corresponding query likelihoods are returned.

For feature extraction with a deep learning approach, we have used a single frame classification method with Resnet50 pretrained on imagenet dataset on top of 4 more layers used to generate feature vectors for individual frames of the videos. On top of Resnet50 architecture, we use 1 layer for average pooling of size 7*7, then a dense layer of size 1024, then a dropout layer with 0.5 rates. In the end, there is an output layer with a length of 5 for each class.

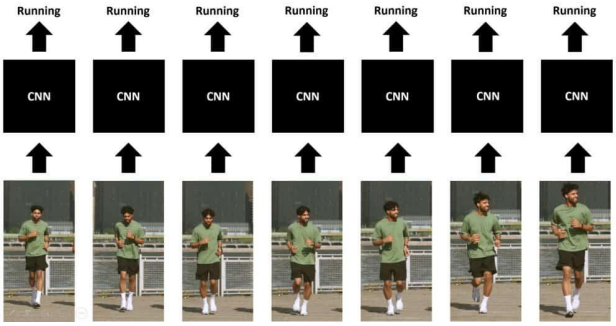


Figure 2: Single Frame classification source:https://learnopencv.com/wp-content/uploads/2021/01/Single-Frame-CNN-Architecture-1024x553.jpg

We will use the train set for training our model with labels belonging to 5 classes (music, vehicles, travel, food, animation). Using the ImageDataGenerator class, we have sampled frames in each video and have used a train/validation ratio of 0.25. For optimizers, we have used SGD; for loss, categorical cross entropy is used; and for metrics, we use accuracy.

Once trained on these five classes, we will use a test set of 50 different videos, with 10 belonging to each class and 5 query videos 1 from each class. For performance considerations, we have compressed our videos to 5 frames per second, and we are only considering 10s of clips for testing. However, we have trained on full-length videos after compression to 5 fps on the training set.

Once we have transformed our videos using our model, we will use the output of the Dense layer of length 1024 for each frame of the video and will store it as an inverted index.

Similarly, transformations are applied to the query video, and similarity is calculated by averaging cosine similarity for each video on the query vector. In our case, the dataset is small, and to maintain higher accuracy; we are not averaging out feature vectors of multiple frames. But averaging out feature vectors or clustering can be used to improve performance for a larger dataset that is in the future scope. These cosine similarity values are used for ranking videos, and ranking is based on the descending order of these values.

Once we have the ranking, we can retrieve the top k results as an output.

Now for a combined model that considers the rankings from the query likelihood model and single frame classification model. Text is processed by the query likelihood model, and video id is processed by the single frame classification model, and both models rank the whole dataset. Those rankings scores are normalized to [0,1] and then averaged out, giving equal weights to both models. These scores are then used to rank the video files.

Empirical results

For the text model, the accuracy will be calculated separately for the bigger 7010 video set and the 50 video set that works with both the text and video models. The reason for this is two-fold. We want to show the accuracy of the 7010 video set because that set has far more videos. In a hypothetical example with a text query simply consisting of the word "soldiers," the query on the 50 video set will return the 4 videos containing soldiers and 6 non-relevant videos because the smaller 50 video dataset doesn't have 10 videos with soldiers. The bigger dataset will thus be better at representing the text model accuracy. The reason we run the text model on the smaller 50-video dataset is so that we can later see how the combined model affects the accuracy scores. Results are shown in Table 2 and 3 :-

Query	Precision @ 10 on Test set
'soldiers at war'	0.9
'car driving'	1
'people''	1
'music'	0.9
'mountain'	1

Table 2: precision at 10 values for 5 text-based queries on the large 7010-video dataset

Query	Precision @ 10 on Test set
‘soldiers at war’	0.4
‘car driving’	0.6
‘people’‘	1
‘music’	0.9
‘mountain’	0.4

Table 3: precision at 10 values for 5 text-based queries on the small 50-video dataset

The lower accuracies in the smaller 50 video dataset can be explained by the small dataset. As written above, there simply aren’t that many videos of “mountain” in the smaller dataset. The videos there do show up as top-ranked in the output, however.

The story with the larger dataset is slightly different but also more interesting. Here the accuracies are very high because as long as the subject is general enough, the dataset will have at least 10 videos. However, the rather rudimentary Query Likelihood model can miss the context of the queries, leading to imperfect accuracies. For example, the case of ‘music.’ A user looking up ‘music’ on a video dataset is probably expecting to see someone playing music or just, in general, music coming through their speakers. However, the top-ranked result for the ‘music’ query on the big dataset is a snippet of someone ranking music clips, but they don’t play the music from said clips. Thus the top-ranked result for music is about music but doesn’t have any music in it. This issue could be fixed with a more comprehensive text model that can account for the query context.

For our image frame classification model, we have used Precision @10 for 5 video query examples of 5 different classes for relevance judgments. Results are shown in Table 4

Topic	Precision @ 10 on Test set
Music	0.5
Animation	0.3
Food	0.4
Travel	0.2
Vehicles	0.3

Table 4: precision at 10 values for video frame classification model

So Average Precision for all of these 5 queries is 0.34. Now the probability of random guess would result in a baseline accuracy of 0.2 because of 5 classes in the dataset. So, this model performs better than a random guess. Given the sufficient amount of resources to train on a larger dataset of videos or by increasing the number of frames per second in the videos, we can reasonably expect an increase in the accuracy of the retrieval system. Since we have a small dataset of videos, we can rank the entire dataset in a reasonable time. So, we used ranked lists to organize the results.

For larger datasets, clustering and comparisons with cluster centroids before ranking individual elements can be performed to increase the performance of the retrieval system with reasonable change degradation in accuracy.

For the combined model we are using Precision @ 10 for evaluating the performance of the model. Results are shown in Table 5

Topic	Precision @ 10 on Test set
Music	0.5
Animation	0.5
Food	0.6
Travel	0.4
Vehicles	0.5

Table 5: precision at 10 values for combined model

We can see that the combined model has average precision @ 10 as 0.5 that is higher than the frame classification model but lower

than the query likelihood model. However, it is still higher than random guess probability and gives relevant results.

Conclusions

In conclusion, we can say that this approach of building separate models for processing different types of information can be beneficial.

So, this shows that individually processing textual and visual information on a distributed system on different models is feasible. Scalability on large datasets is a plus point here relative to one model that gets inference from all types of data together and might need retraining when new data is introduced. In this approach, different models are trained on different sets or types of data and their results can be fine-tuned to meet the required objectives. Although, this can introduce bias while normalizing results across different systems, which can be a plus or a negative depending on the task at hand. So, this is not a one-stop solution, and some models such as visual transformers with text encoding can outperform this approach, however, for now, they do not offer the same flexibility and ease.

If we go over the empirical results, we will find that the text model on the bigger dataset is outperforming results on a smaller 50-video dataset, and that is expected for larger amounts of terms present in the first case for the query likelihood model. However, we can see that single-frame classification is not giving us relatively good results, which can be attributed to a lack of temporal information or activity detection inside the video in single-frame classification. This can also be attributed to differences in relevance judgment for both models. If we consider the features or information used by both models, then the text model will perform better on categories defined lexically, while deep neural networks will give more preference to the similarity in raw data of the frame of video.

The text model was rudimentary, but it worked well for simpler queries. The text model would, however, struggle with any more complicated or query-based search, because it is simply looking for words from the query showing up in the video description. The inability to use context or similarity in any way means an example query of ‘tank’ will not lead to footage of ‘soldiers’ or ‘war’ if those videos do not have a tank in them. For this model, a video of soldiers at a parade would be just as likely to show up for the query ‘tank’ as a video of a mountainscape, even though one is far more related to tanks than the other. However, for simpler queries where the user knows what they want to see and searches for it directly - the model works quite well.

Another issue with the text model is the lack of spell check or word similarity. This means that one misspelling in a smaller query can make it useless. As an example, during the testing of the model we made a query ‘cooking chicken’, except the word chicken was misspelled, so the query was returning cooking videos but they weren’t related to chicken. A more complex method for comparing document and query word similarity might have avoided this issue.

The scope for these solutions in the future is high as they can be modified easily to use any other type of ranking model. Better features can be extracted from video data by considering temporal aspects in it. Visual transformers or visual aggregation techniques require more computational power and can be used in tasks requiring high precision. While our distributed approach can be used in tasks such as web-based content retrieval with search engines where information in metadata such as location and lexical information associated with data is also extremely relevant. Furthermore, to increase performance clustering can be applied to find common centroids of clusters and queries can be first matched with cluster centers before ranking the whole dataset. As observed during training our current model will generalize better and improve over the size of the dataset, however, will require more computational power in training.

SECTION 2

Sample queries, Narratives & relevance judgments

A sample query for our model would be a string describing the contents of the video or the video itself. The string can be sentences, phrases, or single English words. Since our data are divided into 20 categories containing videos with different themes, the query can describe an object or an action.

A sample query would be: “Car on a screen.” The user would want to use this query as input

to our model to find videos containing this element.

Another query would be a 10s video containing a dog dancing. The user would want to use this video to search for similar videos in our dataset.

The judging criteria for the search would be the accuracy and ranking of returned videos. For example, our model returns 10 videos per the user's query, and 2 of them are ranked at the top of the return list and are considered to be related to what he/she wanted. The text-video model is ranked by the naive Bayes query likelihood, and the video-video model is ranked by the cosine similarity of the 2 videos.

In the following, we annotated several example queries through our model, and since we are testing our model against a small slice of data containing only 50 videos, the "accuracy" would not be too high; we are testing if our models return the correct result.

```
For query: " software on computer screen " the top 10 videos are:
1: video598 with score:-15.431759344768501
2: video402 with score:-17.43536290128542
3: video325 with score:-19.226683396095773
4: video185 with score:-19.397685816700868
5: video326 with score:-19.585179058971455
6: video194 with score:-19.782042915113955
7: video544 with score:-19.972703541160943
8: video751 with score:-19.990491114447515
9: video496 with score:-20.016822965944204
10: video171 with score:-20.018759766306466

For query: " group of women singing " the top 10 videos are:
1: video185 with score:-15.52641085939531
2: video207 with score:-15.761276439554726
3: video751 with score:-17.502420499321214
4: video194 with score:-18.124750324979182
5: video645 with score:-18.31481575422639
6: video216 with score:-18.83388546648609
7: video329 with score:-19.175965738885544
8: video250 with score:-19.187721983141152
9: video326 with score:-19.22675384218155
10: video214 with score:-19.29291462690432
```

```
For query: " stylish movie theatre " the top 10 videos are:
1: video171 with score:-15.227043923071054
2: video149 with score:-15.305478588123751
3: video152 with score:-15.708068324663035
4: video544 with score:-16.107800064406142
5: video496 with score:-16.232055374746707
6: video386 with score:-16.549347690172
7: video325 with score:-16.628192210030328
8: video214 with score:-16.695332513354167
9: video227 with score:-16.733034369088966
10: video207 with score:-16.74238626262575

For query: " harvesting rice " the top 10 videos are:
1: video575 with score:-7.648076324523145
2: video325 with score:-11.151403940783108
3: video214 with score:-11.196164142998999
4: video496 with score:-11.202477460721685
5: video227 with score:-11.221298713488864
6: video152 with score:-11.227533309180053
7: video207 with score:-11.227533309180053
8: video250 with score:-11.288833103663698
9: video255 with score:-11.288833103663698
10: video315 with score:-11.288833103663698

For query: " coocking chcicken " the top 10 videos are:
1: video325 with score:-11.432527563748867
2: video214 with score:-11.47728776596476
3: video496 with score:-11.483601083687443
4: video227 with score:-11.502422336454623
5: video152 with score:-11.508656932145813
6: video207 with score:-11.508656932145813
7: video250 with score:-11.569956726629458
8: video255 with score:-11.569956726629458
9: video315 with score:-11.569956726629458
10: video660 with score:-11.569956726629458
```

Figure 3: Retrieved results of the Naive Bayes text-based model.

Query	Description of Video	Top 2 retrieved video
Software on computer screen	1. The interface of a software. 2. A game featuring Disney characters	 

group of women singing	1. Group of women singing and dancing. 2. “Do you hear the people sing”	 
stylish movie theatre	1. A video showcasing a cinema. 2. Behind the scene of Mad Max 4.	 
harvesting rice	1. People harvesting in a rice field. 2. Music video showing people partying on a beach.	 
cooking chicken	1. Music video showing people partying on a beach. 2. Group of men cooking and tasting food.	 

Table 6: Retrieved videos performed by Naive Bayes text-base mode, and their corresponding description.

For video query, we have the following results:

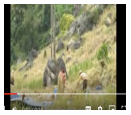
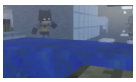
Query Video	Query Likelihood	Top 2 Retrieved videos	Retrieved Video Description
 man speaking while driving	1. 0.67 2. 0.663	 	1. Man singing rock music. 2. Two men singing and playing guitar
 people harvesting in rice field	1. 0.785 2. 0.758	 	1. Group of people walking in the forest. 2. Man wearing boots walking on the snowy road.
 People cooking chicken and broccoli	1. 0.653 2. 0.631	 	1. “Do you hear the people sing” 2. Two men are singing and playing guitar
 The interface of a software	1. 0.782 2. 0.767	 	1. Minecraft batman 2. Music video showing people partying
 Group of women singing and dancing	1. 0.747 2. 0.74	 	1. Two men are singing and playing guitar. 2. Band playing guitar in a concert.

Table 7: Retrieved videos by ResNet video-based search model, and their corresponding description

References

Training & Testing data, Vasili Ramanishka, 2017. URL: shared (mediafire.com)

Code Repository

Github:-**https://github.com/shashibyadav/Video_Clip_Retrieval**

Google Drive:-
https://drive.google.com/drive/folders/1pX1NRShO9WjB4_p0PxRXew3jZWVLXOFZ?usp=share_link