Efficient Video Retrieval with Advanced Deep Learning Models

Tan Luan Huynh*
luanhuynh2001@gmail.com
Ho Chi Minh City University of Technology, VNU-HCM
Ho Chi Minh City, Viet Nam

Minh Toan Dinh*
toandinh6501@gmail.com
Da Nang University of Science and Technology
Da Nang, Viet Nam

Hong Linh Pham*
phlinh0410@gmail.com
Da Nang University of Science and Technology
Da Nang, Viet Nam

Le Hoang Nguyen*
lehoangnguyen510@gmail.com
Institut National des Sciences Appliquées de Rennes
Rennes, France

Tuan Manh Hung Vo*
hungvtm00@gmail.com
Da Nang University of Science and Technology
Da Nang, Viet Nam

ABSTRACT

Video retrieval is the process of finding specific video content in a large database. This is a crucial challenge in the age of digital multimedia. This article proposes a new approach to video retrieval using advanced deep learning models to extract features and perform retrieval tasks based on those features. Our method combines multiple feature extraction methods, including keyframe extraction, OpenAI CLIP [7] feature extraction, object detection, and automatic speech recognition (ASR). We use BERT [3] embeddings to encode these transcripts and store them in JSON and binary file formats. Our system achieves remarkable results in indexing and retrieving videos based on their visual, audio, textual, and contextual attributes. Our system can also retrieve videos based on either a single text description or multiple text descriptions of a sequence of events. We conducted extensive tests on diverse video data from Ho Chi Minh City AI Challenge 2023 competition organizers to validate the effectiveness of our approach. The results demonstrate that our proposed system is superior to other methods in terms of both retrieval accuracy and speed, making it highly suitable for real-time applications.

CCS CONCEPTS

• **Computing methodologies** → **Visual content-based indexing and retrieval**; • **Information systems** → **Video search**; **Top-k retrieval in databases**; **Information retrieval diversity**; **Search interfaces**.

KEYWORDS

interactive video retrieval, text-based search, event retrieval, multimedia retrieval system, automatic speech recognition, object detection

*All authors contributed equally to this research.

SOICT 2023, December 07–08, 2023, Ho Chi Minh, Vietnam

ACM ISBN 979-8-4007-0891-6/23/12…$15.00
https://doi.org/10.1145/3628797.3628995

ACM Reference Format:
Tan Luan Huynh, Minh Toan Dinh, Hong Linh Pham, Le Hoang Nguyen, and Tuan Manh Hung Vo. 2023. Efficient Video Retrieval with Advanced Deep Learning Models. In *The 12th International Symposium on Information and Communication Technology (SOICT 2023), December 07–08, 2023, Ho Chi Minh, Vietnam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3628797.3628995

1 INTRODUCTION

In the age of information abundance, the management and retrieval of multimedia content, particularly videos, has become increasingly challenging. Videos, as a dynamic amalgamation of images, audio, and textual information, offer a vast repository of knowledge, entertainment, and insights. However, video content's sheer volume and diversity pose significant challenges for efficient exploration and retrieval.

Video Text Retrieval (VTR) is a rapidly emerging field that aims to bridge the gap between visual and textual content in videos. This is motivated by the profound impact of video as a communication and information-sharing medium. As videos continue to increase across various online platforms, including social media, educational resources, news outlets, and surveillance systems, accessing and organizing the textual content embedded within these videos has become increasingly pressing.

This paper presents a comprehensive system for Video Text Retrieval (VTR) that leverages cutting-edge technologies from computer vision, natural language processing, and machine learning. Our system aims to streamline the extraction, indexing, and retrieval of information from videos, enabling users to access and interpret multimedia content with unprecedented ease and precision.

In other words, our system aims to make it easier and more accurate for users to find the information they need in videos. We use advanced technologies to extract, index, and retrieve information from videos based on textual queries. We also optimize search results by filtering videos to include only those that contain the objects that users want, or by searching based on the video's audio. Our system offers diverse applications, including assisting users in locating specific details such as events or activities taking place in videos. It also enables content-based video searches, allowing users to find videos related to specific topics like news stories or movie scenes. Additionally, the system helps users discover videos

aligned with their interests, such as those featuring their favorite sports teams or hobbies.
The main contributions of our work include proposing a comprehensive system for Video Text Retrieval (VTR) that leverages state-of-the-art technologies from multiple disciplines; providing a detailed account of the system's components and the methodologies and technologies used for each task and evaluating the system on a benchmark dataset and demonstrate its effectiveness in retrieving videos based on text-based queries.

2 RELATED WORKS

2.1 Contrastive Learning

Contrastive learning stands as a pivotal method in the field of machine learning, rooted in the comparison of objects or data points within a representation space. The objective of contrastive learning is to ensure that similar objects possess representations that are close in the representation space, while dissimilar objects have representations that are distant from each other. The operation of contrastive learning typically involves the creation of training data pairs, comprising a positive sample and a negative sample. The positive sample consists of data points that are similar or belong to the same class, while the negative sample comprises dissimilar data points or data points from different classes. Contrastive learning models are trained such that the distance between representations of positive samples in the representation space is minimized, while the distance between positive and negative samples is maximized. This aids the model in learning representations conducive to distinguishing between objects or data points effectively.

2.2 Development and Applications of Contrastive Learning

Contrastive learning has undergone substantial development, initially emerging in the realm of natural language processing. A notable example is the Word2Vec [6] model, where the contrastive method was applied to learn vocabulary representations by predicting words from the surrounding context. Simultaneously, in the field of computer vision, models like SimCLR [1] and [4] also utilize contrastive learning to train image representations. These representations find widespread applications in tasks such as image classification, object detection, and facial recognition.
In the past decade, contrastive learning methods have become increasingly popular, particularly with the advent of deep learning models and related representation learning applications. t

2.3 Models Utilizing Contrastive Learning for Image Retrieval Tasks

Numerous models employ contrastive learning methods to learn image and text representations in the context of content-based image retrieval. For instance, the VirTex [2], ICMLM [9], ConVirt [11], and CLIP [7] models all employ this approach, but they differ in network architecture, training scale, and learning objectives. VirTex [2] and ICMLM [9] focus on text-based image retrieval and vice versa. ConVirt [11] is capable of achieving multiple learning objectives simultaneously and has a smaller scale. Finally, CLIP [7] is a special-purpose multi-task model with a broader range of pre-training tasks, including OCR, geo-localization, action recognition, and many others, with a multi-modal architecture and a larger scale.

3 METHOD

3.1 System

The system consists of two main parts: feature extraction and data retrieval based on the extracted features.

3.1.1 Feature Extraction.

The essence of videos consists of a sequence of consecutive frames describing events recorded in the video. However, information in videos appears in various forms. To maximize the amount of information extracted from a video, we perform feature extraction for each type of information. (Figure 1)
Firstly, we extract keyframes from the video. These keyframes depict an event occurring at the specific moment of the keyframe. We employ the Transnet [10] model to detect scene transitions in the video. For each detected scene, the first, middle, and last frames are retained and considered representative keyframes for the scene. These keyframes are stored in the database to facilitate the retrieval process.
Next, we detect objects present in each scene using the extracted frames. Information about object types, quantities, etc., is saved for future retrieval purposes.
We also explore human speech data in the video to perform additional feature extraction. Firstly, all audio in the video is converted into raw text. Subsequently, we preprocess the text data using common Natural Language Processing (NLP) techniques such as punctuation removal, special character removal, stopwords elimination, etc. To extract features from the textual data, we employ BERT word embeddings to encode all sentences in the text into multidimensional vectors. These vectors are saved to serve the retrieval process.

3.1.2 Event Retrieval (Feature-Based Retrieval).

Having explored and extracted features from various types of data, we designed a retrieval system incorporating multiple methods suitable for each data type. (Figure 2)
For keyframes, we perform queries through two approaches: text-based retrieval and image-based retrieval. To retrieve events using text, users input a textual query into the system, and the results returned are keyframes representing that event. Here, we use the CLIP model for retrieval tasks. Once the keyframes are obtained, users can choose any frame for image-based retrieval to find similar frames (image retrieval).
For textual data, we perform queries using semantic search methods. The user-input textual query is encoded into vectors through BERT word embedding and matched with feature vectors in the database using cosine similarity to find related texts. After matching, the indexes of the vectors are mapped to corresponding texts and returned to the user.
To make searching by object more convenient, we also perform object detection on each keyframe, so that we can more easily find

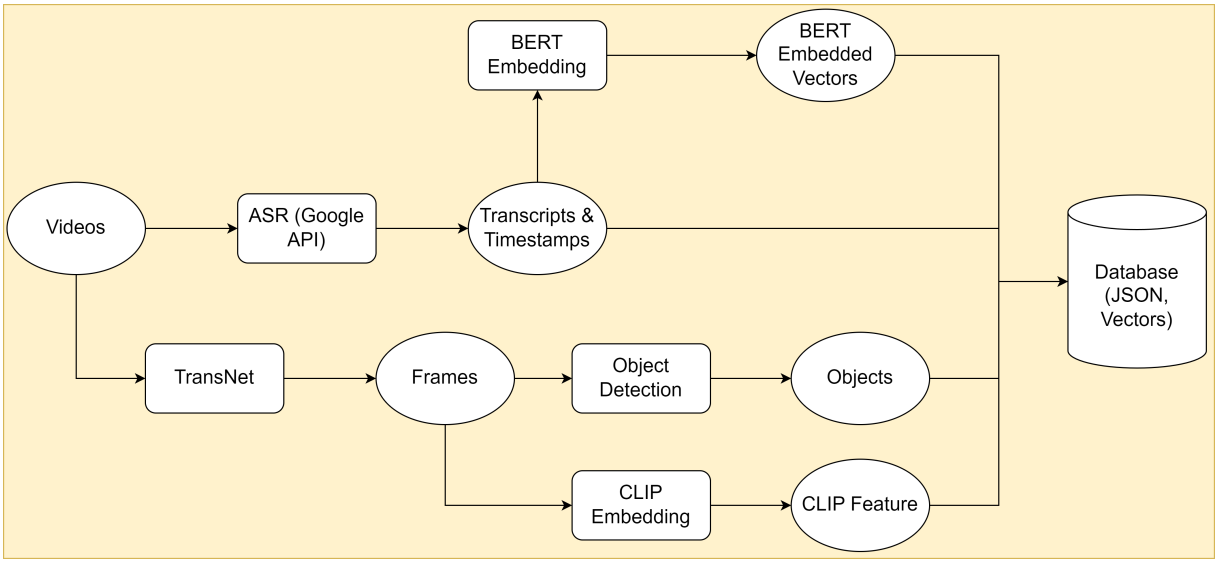

Figure 1: Our features extraction pipeline

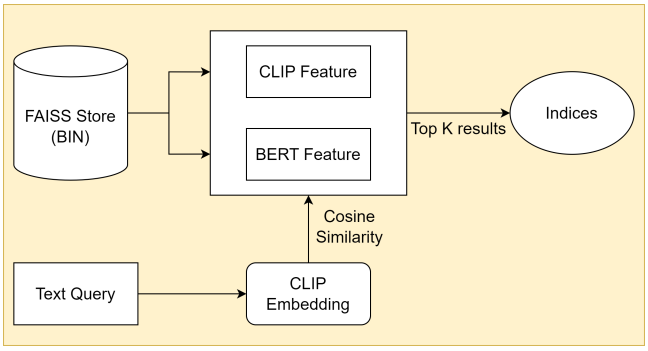

Figure 2: Our video retrieval pipeline

the frame with the described text and the objects that are likely to appear.

In some cases, we need to find multiple consecutive frames depicting an action instead of just one frame. To address this need, we developed the "sequence search" feature. The input to the feature includes two sentences describing the first and last paragraphs of the clip we want to find, and the output is a series of frames representing these sentences.

3.2 Contrastive Language-Image Pre-Training (CLIP)

Here is a description of how to apply OpenAI's CLIP to a text-based video retrieval system using a pre-trained CLIP model to extract CLIP features for each image:

Firstly, the data to be prepared must include all the keyframes that were extracted from all the videos in the dataset.

Secondly, the CLIP model is trained on the prepared dataset. The training process can be time-consuming, depending on the dataset's size and the model's complexity. To save time and computational resources, we recommend using a pre-trained CLIP model.

Once the CLIP model is available, it can be used to extract CLIP features for each image in the dataset. CLIP features are a vector of numbers that can be used to represent an image.

Finally, when a user enters a text description, the system will use the CLIP model to encode this text into a vector. The system then uses the Faiss [5] module and cosine similarity to find all keyframes with CLIP features similar to the vector created from the text description. CLIP is a powerful machine learning model that can be used to improve the effectiveness of text-based video retrieval systems. By using CLIP and Faiss [5], the system can find keyframes that match a text description more accurately and quickly.

3.3 Object detection

For object detection, we used Faster R-CNN [8] on each keyframe to identify all objects in the frame. This process included detecting the class, determining the bounding box, and calculating the probability of belonging to this class. The results were then stored in the database.

When in actual use, the situation is that we can describe the video part by text, and we know what object would be in this part. We need to give the system the text, and these objects. The system first uses the provided text to handle the text retrieval task and outputs a list of proposal frames, then the system will filter this list to take the frames that have all objects we provided.

3.4 Automatic Speech Recognition (ASR)

We store all data, including the original data (videos) and the data generated during the ASR task (audio files, transcripts, vectors). However, only the transcripts and the embedded vectors are stored in the database and indexed for retrieval tasks.

The transcript is stored as a JSON file containing a list of transcripts for the videos. In the list, each item is a key-value pair with the key being the video ID and the value being a list of transcript-timestamp

pairs. The vectors representing the transcript for the video will be stored as a binary file.

3.5 Facebook AI Similarity Search (Faiss)

We apply Faiss (Faiss Facebook AI Similarity Search) to our text-based video retrieval system:

Indexing with Faiss. Faiss is used to create an index structure for the feature vectors extracted from a large multimedia database. Features (e.g., embeddings) are extracted from the data and added to the Faiss index. The index structure allows for efficient storage and retrieval of feature vectors. This step significantly improves the speed of similarity searches in the database.

Searching with Faiss. Faiss is used for performing similarity searches, both for images and text inputs. It provides methods for searching similar images based on image queries and text queries. Faiss efficiently retrieves the nearest neighbors (similar items) to the input vectors, enabling fast and accurate search results.

4 EXPERIMENTAL RESULT

4.1 Data

4.1.1 Data description.

The data used in this study comprises videos, keyframes, map keyframes, objects, and metadata:

- Videos: These consist of news video clips and documentary-style programs, with an average duration ranging from 20 to 40 minutes per video. A total of 1211 videos were utilized in this study.
- Keyframes: Keyframes are individual frames extracted from video files. These frames are selected at intervals to ensure that adjacent frames are dissimilar.
- Map Keyframes: These are .csv files that store corresponding mapping information for keyframes, associating them with the sequential frame numbers within their respective videos.
- Metadata: The metadata contains information about the videos, including details about the video's author, the video's file path, descriptions, keywords, start and duration times for keyframes, and transcripts of the video content during those time intervals. The transcript information will be used for keyframe retrieval through semantic search.
- Objects: The objects component includes information about objects present in each keyframe, such as labels and bounding boxes. This information will be employed for keyframe retrieval through object detection.

4.1.2 CLIP features.

In this section, we provide a detailed account of our data processing procedures, encompassing the creation of two pivotal components essential for our experimental endeavors.

CLIP Embedding. We commence our data processing pipeline with the computation and archival of CLIP embeddings. These embeddings represent a fusion of textual and image data, rendering them multi-modal. Their fundamental characteristic lies in their ability to encapsulate the semantic information shared between images and textual descriptions. These embeddings serve as repositories of semantically rich insights about both images and text, enabling the efficient execution of similarity-based search operations.

JSON Metadata. In tandem with CLIP embeddings, we generate a JSON metadata file, aptly named "keyframes_id.json." This metadata file plays a foundational role by establishing a seamless mapping between image indexes and their corresponding source images.

Leveraging the FAISS Library. To empower rapid and scalable similarity-based searches, our methodology leverages the capabilities of the FAISS library. FAISS is meticulously designed for the execution of efficient similarity searches within extensive datasets. Within this framework, we import the CLIP embeddings, meticulously stored in binary format (.bin), into a FAISS index. The choice of index type, such as L2 for Euclidean distance, is discerningly determined based on the specific requirements of our experiments.

Image-Based Search. When a user initiates an image-based retrieval, the FAISS index springs into action, adeptly retrieving the closest CLIP embeddings. These embeddings are subsequently linked to the actual image files, guided by the information encapsulated within the JSON metadata. This meticulous mapping process ensures that each search result can be unequivocally associated with its source image, simplifying the interpretation of results.

Text-Based Search. In the realm of text-based search experiments, textual descriptions serve as the query input. The FAISS index, still housing the CLIP embeddings, assumes a pivotal role. It excels in identifying the most semantically relevant images based on textual content. Following the retrieval of results, often presented in the form of indexes or IDs, the JSON metadata file proves indispensable in mapping these outcomes to the actual image file paths.

Our meticulously devised data processing methodology, underpinned by CLIP embeddings stored in .bin files, FAISS indexing, and JSON metadata mapping, presents a versatile and robust solution for the retrieval and interpretation of visual content, spanning both images and text. This methodology holds significant promise across diverse applications, ranging from image retrieval to recommendation systems and beyond.

4.1.3 Object detection.

We first use a computer vision model called Faster R CNN to detect objects in all keyframes, and we store the results in JSON files. When a user wants to search for a video clip by text, they use the "text retrieval" module. However, sometimes users may want to find video clips that contain specific objects. To support this, the system allows users to filter frames based on the presence of objects with a specified confidence threshold.

The system also provides two ways to filter objects: and and or. The and option returns all frames that contain all of the classes that the user inputs. The or option returns all frames that contain at least one of the classes in the user's list of classes.

4.1.4 Automatic Speech Recognition.

In this section, we describe our ASR process. Figure 3 shows our ASR pipeline

Firstly, we store all data, including the original data (videos) and the data generated during the ASR task (audio files, transcripts, vectors). However, only the transcripts and the embedded vectors are stored in the database and indexed for retrieval tasks.

The transcript is stored as a JSON file containing a list of transcripts

for the videos. In the list, each item is a key-value pair with the key being the video ID and the value being a list of transcript-timestamp pairs. The vectors representing the transcript for the video will be stored as a binary file.

4.2 System implementation detail

Our image retrieval system has been successfully implemented with a comprehensive backend and frontend, allowing users to search for images efficiently using various methods.

The backend of the system is built in Python, where we process input data from the front end. We utilize the Faiss library to construct a search model and perform image searches. Results from the backend are passed through the Flask framework, where API endpoints are set up to handle search requests from the front end. The output from the API is an HTML webpage.

On the frontend side, the user interface is constructed by using only one HTML page with CSS, and JavaScript. The webpage is structured with two key components: the search bar and the image display area. The search bar enables users to easily select a search mode, set the number of images to retrieve and initiate searches based on their preferences.

The system supports five main search modes:

- **Text Search mode** is the default mode of our system, allows users to enter a query to search for images. They can also optionally apply object filters to narrow down the results to only include images that contain specific objects. Users can also set a minimum confidence threshold of each object to fine-tune search outcomes. The system also provides two "operators": "and" and "or". The user uses the "and" operator when they want the system to keep only keyframes that contain all of the objects provided by the user. Conversely, if the user requires the system to accept at least one of the objects proposed by the user to appear, they select the "or" option.
- **Sequence Search mode** enables users to input two sequences to search for consecutive images within those sequences.
- **Image Search mode** allows users to enter a keyframe ID to receive the corresponding YouTube link for that image and other related images that are also in the same video. The Submit button is used to send selected data or search results to the backend for processing.
- **Transcript Semantic Search mode** allows users to enter a query and return a list of transcript segments, timestamps, and corresponding videos that are relevant to that query. This mode allows users to search for videos based on their transcripts
- **Image Retrieval mode** allows users to find keyframes that are similar to a given keyframe. The system uses Faiss to compute the cosine similarity of the given keyframe's CLIP features vector to the CLIP features vectors of all keyframes in the database. The keyframes with the highest cosine similarity scores are returned as the results.

The combination of a powerful Python backend and an interactive user interface on the front end has created a versatile image retrieval system, making it easy for users to access and search for images using a variety of methods.

4.3 Experimental query

We provide some special examples of our system's efficient video retrieval, using text-based search (to search for videos by providing text descriptions), combined with features extracted from OpenAI CLIP models, object detection, transcript semantic search and sequence search for some special query cases.

When a user wants to use the system we propose to search for a video, they first describe the video in text and enter it into the system. The system will then return a list of promising keyframes (by default, the top 300 most relevant keyframes will be returned, but the user can customize this number to search for more). Users can click on any keyframe to view detailed information about the keyframe and the short video containing that keyframe. However, there are some cases where the provided text is not enough, and it can be time-consuming to search. At this point, it will be necessary to use the additional modes that we propose in Section 4.2:

- Search by video transcript: The user enters text describing the transcript that may appear, and the system will return possible videos.
- Object filtering to keep only keyframes that contain objects that the user suspects will appear in the video.
- Sequence search allows users to search by entering multiple query sentences, describing multiple scenes that occur sequentially in the video.
- If a user wants to search for keyframes that are similar to a given keyframe, they can use the image retrieval mode to retrieve them.

We use the dataset containing a large number of videos provided by the organizers of the Ho Chi Minh City AI Challenge 2023 to demonstrate our system's capabilities.

4.3.1 Queries need to apply ASR Semantic Search.

To improve the performance of our video search system, we used BERT to retrieve text data from videos. This is important because our main retrieval model, CLIP, can sometimes fail to detect events in poor lighting conditions or when the objects of interest are dark in color.

To demonstrate the flexibility and complementarity of switching between keyframe-based (CLIP) and text-based (BERT) search, consider the following example. A user might submit the following query: "The video shows a black car driving with a person clinging to the hood. The car crashes into a motorbike. After a while, two people wearing green shirts run up from behind."

CLIP would likely have difficulty detecting the important objects in this video, such as the black car, the person clinging to the hood, and the two people wearing green shirts. This is because the video is taking place at night in poor lighting conditions.

To improve our chances of finding a matching video, we paraphrased the user's textual query with some possible contexts like traffic accidents or scenes in a movie. We then used BERT and Semantic Search to retrieve videos that matched these contexts. The results showed that we were able to find a matching video in which a policeman clings to the hood of a car while patrolling the road and tries to stop the car.

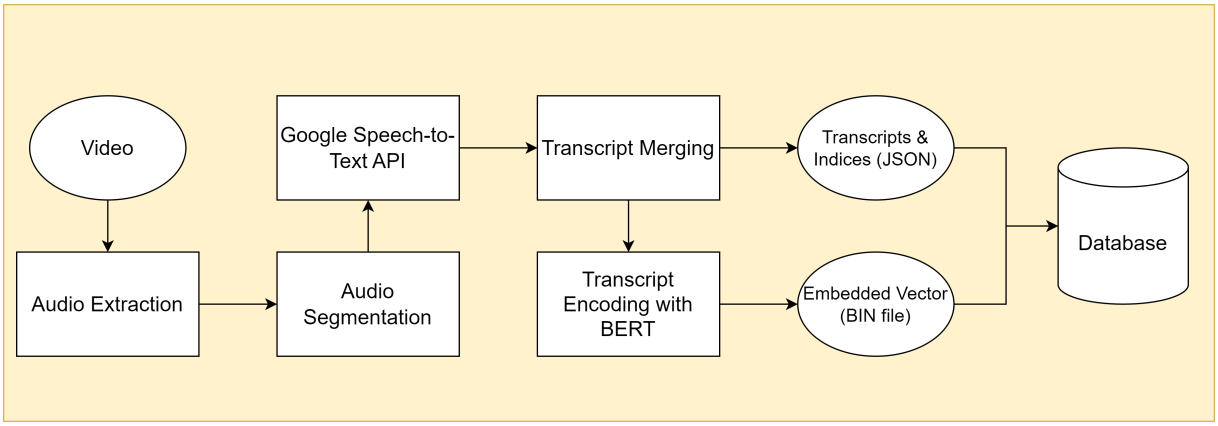

Figure 3: Our Automatic Speech Recognition Pipeline

Let's consider another example. If we search for a video with a long and complex description, such as "Scene of a car driving onto the sidewalk and hitting 4 motorbikes. A person was sitting nearby buying things from a stall on the roadside. It ends with a scene of someone running out of the house to see the situation," it will be difficult to find the correct video using just keyframes. This is because many accidents on the road involve both cars and motorcycles, and it would be time-consuming to check all of the keyframes in the database (Figure 4). To make this easier, we can think about how the news would report on this event. We can then use a query for Transcript semantic search, such as "A traffic accident happened on a road caused by a car, resulting in the car crashing onto the curb and colliding with 4 other motorbikes." Semantic search will find the keyframes that have the most similar transcript to the query. This will reduce the number of keyframes that we need to check, and make it more likely that we will find the correct video.
These are just a few examples of how we can use BERT and Semantic Search to improve the performance of our video search system. By incorporating different contexts into our queries, we can increase the chances of finding matching videos, even when the objects of interest are difficult to detect.

4.3.2 Queries need to apply Sequence Search.

The effectiveness of CLIP (Text Search mode) is limited when the query describes multiple segments of the same video. In this case, CLIP may only return images that are most similar to the query description. To address this limitation, we propose a sequence search approach that considers the temporal relationship between query segments.
For example, the query "Scenery of a mountain range covered in soil and sand, with cranes and dump trucks operating, followed by an image of workers discussing" may return images of mountains or trucks, or workers (Figure 5). However, if we use sequence search with sequences 1 and 2 as follows, we can obtain more accurate results (Figure 6):

- Sequence 1: Scenery of a mountain range covered in soil and sand, with cranes and dump trucks operating
- Sequence 2: Workers discussing

In this case, the sequence search approach will return images that are sequentially related to the query descriptions in the same video. This results in improved search accuracy and relevance.

4.3.3 Queries need to apply Objects Filter Module.

CLIP is a powerful model for video retrieval. However, it can be difficult to find relevant keyframes when the textual query contains a lot of information and many objects. In these cases, we can use a secondary model, such as an object detection model, to filter out keyframes that do not contain the specified objects.

For example, if we search by the query "Man in red shirt plays musical instrument near a small peach tree, on Tet holiday, in a room", we obtain the results shown in Figure 7 when we use the regular Text Search mode. The results are chaotic because there is a lot of information and many objects in the input query. To improve the results, we can use the object filter module to keep only keyframes that contain both the "Person" and "Musical instrument" objects. This allows us to quickly find the correct keyframe in the first row of Figure 8 (The true keyframe is highlighted). The "And" operator in Figure 8 is used to keep all keyframes that contain all of the chosen objects. The results are selected from the 1000 most suitable keyframes, with an object confidence threshold of 0.3.

5 CONCLUSION

In this paper, we describe a video retrieval system based on various data types to participate in the Ho Chi Minh AI Challenge 2023. The system consists of several retrieval modules and supports complex multimodal queries, including frame-based and text-based queries. We exploit a combination of advanced artificial intelligence techniques to automatically analyze the visual content of the video keyframes and extract objects as well as the relationships between them. Speech-to-text converted text is also considered as additional data for retrieval. All extracted features are converted into specifically designed text encodings, which are then indexed by a full-text search engine. Evaluation through system testing using textual queries shows that the retrieval system works effectively with most queries due to the flexible ability to change the retrieval methods between images and text.

Figure 4: The result of Text Search module: "A car driving onto the sidewalk and hitting 4 motorbikes. A person was sitting nearby buying things from a stall on the roadside. It ends with a scene of someone running out of the house to see the situation."

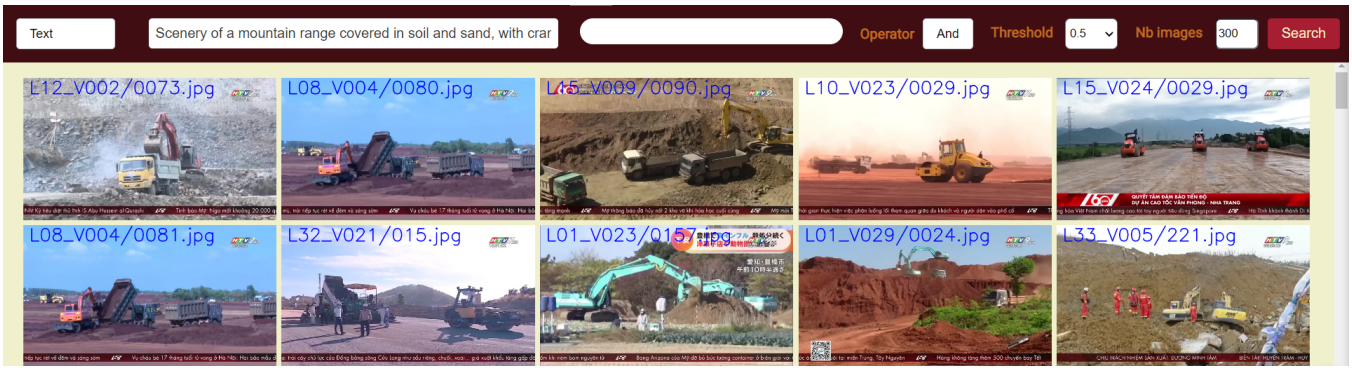

Figure 5: The result of Text Search module: "Scenery of a mountain range covered in soil and sand, with cranes and dump trucks operating, followed by an image of workers discussing"

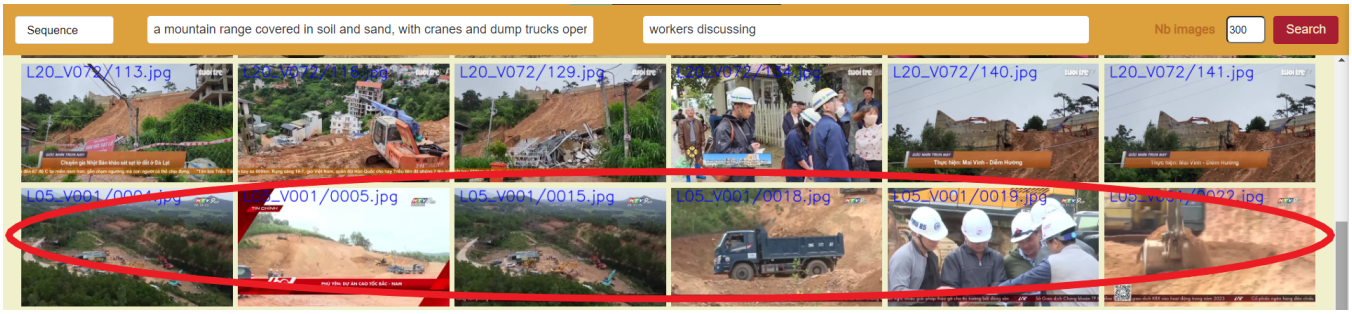

Figure 6: The result of Sequence search. Sentence 1: "Scenery of a mountain range covered in soil and sand, with cranes and dump trucks operating". Sentence 2: "Workers discussing". The keyframes of the correct video are highlighted.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. arXiv:2002.05709 [cs.LG]

[2] Karan Desai and Justin Johnson. 2021. VirTex: Learning Visual Representations from Textual Annotations. arXiv:2006.06666 [cs.CV]

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. arXiv:1911.05722 [cs.CV]

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *CoRR* abs/1702.08734 (2017). arXiv:1702.08734 http://arxiv.org/abs/1702.08734

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV]

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Figure 7: The result of Text Search module: "Man in red shirt plays musical instrument near a small peach tree, on Tet holiday, in a room"

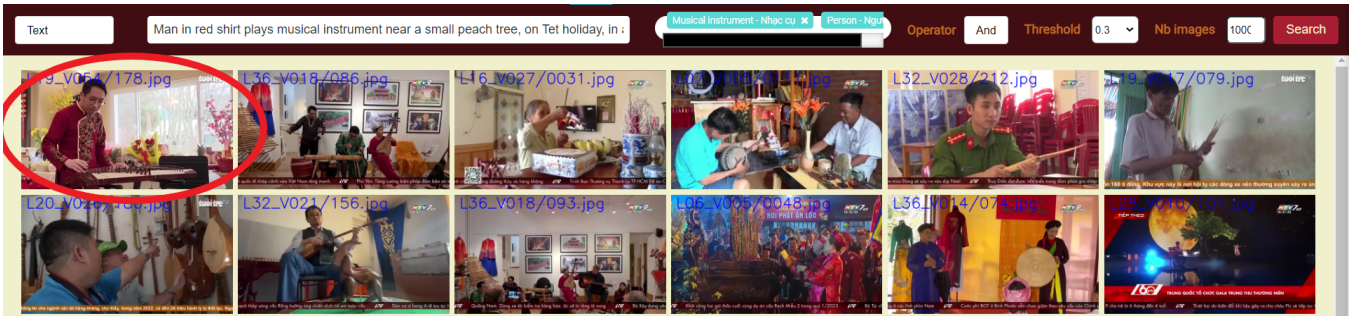

Figure 8: The result of Text Search that applies Object Filter: "Man in red shirt plays musical instrument near a small peach tree, on Tet holiday, in a room", using object filter ("Musical instrument", "Person"), operator "And", search for 1000 keyframes with Confidence threshold is 0.3. The keyframes of the correct video are highlighted.

[9] Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. 2020. Learning Visual Representations with Caption Annotations. arXiv:2008.01392 [cs.CV]
[10] Tomás Soucek, Jaroslav Moravec, and Jakub Lokoc. 2019. TransNet: A deep network for fast detection of common shot transitions. *CoRR* abs/1906.03363 (2019). arXiv:1906.03363 http://arxiv.org/abs/1906.03363
[11] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. 2022. Contrastive Learning of Medical Visual Representations from Paired Images and Text. arXiv:2010.00747 [cs.CV]