PAINE Demo: Optimizing Video Selection Queries With Commonsense Knowledge

Wenjia He
University of Michigan, Ann Arbor
wenjiah@umich.edu

Ibrahim Sabek
Massachusetts Institute of Technology
sabek@mit.edu

Yuze Lou
University of Michigan, Ann Arbor
yuzelou@umich.edu

Michael Cafarella
Massachusetts Institute of Technology
michjc@csail.mit.edu

ABSTRACT

Because video is becoming more popular and constitutes a major part of data collection, we have the need to process video selection queries — selecting videos that contain target objects. However, a naïve scan of a video corpus without optimization would be extremely inefficient due to applying complex detectors to irrelevant videos. This demo presents Paine; a video query system that employs a novel index mechanism to optimize video selection queries via *commonsense knowledge*. Paine samples video frames to build an inexpensive lossy index, then leverages probabilistic models based on existing commonsense knowledge sources to capture the semantic-level correlation among video frames, thereby allowing Paine to predict the content of unindexed video. These models can predict which videos are likely to satisfy selection predicates so as to avoid Paine from processing irrelevant videos. We will demonstrate a system prototype of Paine for accelerating the processing of video selection queries, allowing VLDB'23 participants to use the Paine interface to run queries. Users can compare Paine with the baseline, the SCAN method.

PVLDB Reference Format:
Wenjia He, Ibrahim Sabek, Yuze Lou, and Michael Cafarella. PAINE Demo: Optimizing Video Selection Queries With Commonsense Knowledge.
PVLDB, 16(12): 3902 - 3905, 2023.
doi:10.14778/3611540.3611581

1 INTRODUCTION

With the increased availability and popularity of video databases [1, 7, 8], video selection queries have emerged as a growing area of research interest [2, 3, 5, 6, 9]. These queries are utilized for selecting desired videos that satisfy certain predicates, especially containing target objects detected by neural networks. This kind of query can help video search in consumer-facing systems (e.g., social media platforms, albums in personal smartphones), in analytical systems for skilled users, and in systems for constructing training sets (as with self-driving cars). Due to the large size of existing video databases, it is a common practice to include the LIMIT clause in selection queries in order to limit the size of the returned video set.

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611581

For example, a user might want to search for 12 videos containing motorcycles from a large video corpus. A selection query for this purpose is shown as follows:

```
SELECT * FROM videoCorpus
WHERE DetectedObject = ''motorcycle''
LIMIT 12
```

Here, the DetectedObject field is populated by applying a frame-level object detector to each image in videoCorpus. The current state-of-the-art object detectors are neural networks. A naïve query processing method for this query is to simply scan the video corpus, repeatedly applying the object detector and testing the results until the result set's size meets the LIMIT value. However, the significant number of videos that do not satisfy the predicate and the detector's long inference time would lead to extremely slow processing.

Precomputing and indexing the contents of DetectedObject would allow us to avoid work at query time. Unfortunately, building an index of the complete video content requires processing every frame by the object detector and recording all the object information. Applying the object detector at ingest time or retrospectively does not make a big difference: the quantity of work is enormous in either case. For online media platforms like YouTube, over 500 hours of videos are uploaded every minute [10]. Taking the state-of-the-art model YOLOv7 [13] as an example, the processing rate is only 56 FPS when achieving 55.9% Average Precision. To reduce the index cost, a straightforward option is to sample fewer frames for processing; unfortunately, this means objects that only appear in a small fraction of frames are likely to be missed. Existing techniques cannot achieve a high index quality with a limited index budget. The difference detector method [6] prevents the object detector from processing visually similar frames at index time, but it only works when the video is very static. The specialized neural model method [2, 5, 6] trades the generality and accuracy of a detector for fast inference so that more frames can be processed, but these two features are important for a high-quality index.

In this paper, we demonstrate Paine; a video query system that employs a novel index mechanism to optimize video selection queries. We find that there are semantic-level correlations in videos; a human being can predict most of the video contents after observing just a few frames. Consider the above user who wants to see videos that contain motorcycles as an example. If she were personally surfing videos to find something that contains motorcycles, she might reasonably skip over a video whose first frame is in a kitchen; it almost certainly does not contain motorcycles. But she is likely

interested in watching more of a video whose first frame shows cars on a road, even though she does not yet see a motorcycle, because cars and motorcycles often go together.

Paine leverages an inexpensive but lossy index and commonsense knowledge to prioritize promising predicate-related videos. It consists of three stages. In the offline *Model Preparation* stage, Paine can build two kinds of probabilistic models to infer the missing objects from the lossy index — one is a derived conditional probability formula from knowledge graphs [12], and the other is a regression model learned from videos based on the pre-trained BERT model [4]. In the *Indexing* stage, for any arriving video, frames are sampled and processed by the object detector to create an inexpensive index. This processing rate is adjustable according to the index time budget. In the *Query Processing* stage, when a selection query arrives, based on the lossy index and a commonsense knowledge-integrated probabilistic model, Paine predicts the existence probability of the target object and ranks videos in descending order of this probability. Videos that are likely to satisfy predicates can be processed first, thereby avoiding unnecessary processing of irrelevant videos while ensuring 100% accuracy.

We demonstrate a system prototype of Paine, showing how this new index mechanism performs for video selection queries on YouTube videos, which are typical and important workloads. In this use case, users can specify the target object and the LIMIT value in the query. This demonstration shows the query runtime with two different versions of the index mechanism: one built using knowledge graphs, the other using commonsense knowledge from videos. To reveal how different probabilistic models reduce the processing time, it displays the processed relevant and irrelevant videos that are directly affected by different optimization methods. In addition, users can compare Paine with a SCAN method that uses the same lossy index as Paine.

2 PAINE ARCHITECTURE

Figure 1 depicts the three-stage architecture of Paine. This system allows interaction with non-technical users: users provide video selection queries by specifying the target objects and the LIMIT values; Paine selects the satisfying videos from a video corpus through fast processing and returns them to users. We introduce these three stages in Section 2.1-2.3.

2.1 Indexing Stage

Complex neural network models can compute good-quality detection results for the non-lossy index, but per-frame processing would take quite a long time. In the *Indexing* stage in Paine, an inexpensive lossy index is built. A fraction of frames in each arriving video are processed by an object detector (YOLO9000 [11] in Paine) to construct incomplete object lists; these object lists mapped to videos as key-value pairs compose the index. The processing frame rate for the index is adjustable according to varying index time budgets; by default, the rate is 1 out of 30 frames in Paine.

2.2 Query Processing Stage

When a new query arrives, the system enters the *Query Processing* stage. Algorithm 1 describes this procedure. In line 1, the system computes the probability of observing the target object with a probabilistic model (Section 2.3), conditioned on the fact we have observed the objects that are present in the lossy index. This step requires only a few seconds, which is nearly negligible when compared to object detection. In line 2, videos in the corpus are sorted in descending order of this probability. If the target objects happened to be directly observed during indexing, that video is added to the result set directly in lines 5-8. Videos that are not yet in the result set are then processed in the computed order in lines 9-11. We apply the object detector to each frame in the video to determine unambiguously whether the video contains the target object. Videos that do contain the target object are added to the result set until the set's size has reached the LIMIT number or all the videos have been explored. A high-quality probabilistic model will mean processing fewer videos before returning the result set, yielding less processing time.

Algorithm 1: Query processing

Input: Video corpus $\mathcal{V}$, object detector D, target objects O, LIMIT number k, probabilistic model M, Index $\mathcal{I}$

1 $\mathcal{P} = M(\mathcal{I}, O)$;
2 Sort $\mathcal{V}$ in descending order of $\mathcal{P}$;
3 **repeat**
4 $\quad V_{select} = \mathcal{V}$.getNext();
5 $\quad$ **if** $O \subseteq \mathcal{I}_{V_{select}}$ **then**
6 $\quad\quad resultSet$.append(V_{select});
7 $\quad\quad$ continue;
8 $\quad$ **end**
9 $\quad$ **if** $O \subseteq D(V_{select})$ **then**
10 $\quad\quad resultSet$.append(V_{select});
11 $\quad$ **end**
12 **until** $|resultSet| == k$ or $\mathcal{V}$.hasNext() == False;

Output: $resultSet$

2.3 Model Preparation Stage

The probabilistic model is constructed in the offline *Model Preparation* stage to support fast processing. It predicts the probability that a video contains object X, given that we know object Y exists in the video. We have two different methods for building this model depending on whether historical videos are accessible.

2.3.1 *Knowledge Graph Model.*

Knowledge graphs can describe the relationship between objects, represented by nodes and edges. Paine utilizes ConceptNet Numberbatch [12], a set of word embeddings learned from a knowledge graph ConceptNet, containing general commonsense knowledge. The cosine similarity between word embeddings can be computed for measuring word similarity. When two objects are in the same domain, they are semantically similar and tend to appear in the same video, e.g., a motorcycle and a car. Therefore, for a single target object O and an observed object list $\mathcal{I}_i$ containing only one object in the index, we take the cosine similarity as the conditional probability estimation:

$$P(O|\mathcal{I}_i) := \max\left(\frac{embedding(O) \cdot embedding(\mathcal{I}_i)}{\|embedding(O)\| \cdot \|embedding(\mathcal{I}_i)\|}, 0\right). \quad (1)$$

When there are multiple distinct objects in $\mathcal{I}_i$, denoted as $I_{(i,1)}$, $I_{(i,2)}$, ..., $I_{(i,m_i)}$, Paine estimates the conditional probability $P(O|\mathcal{I}_i)$

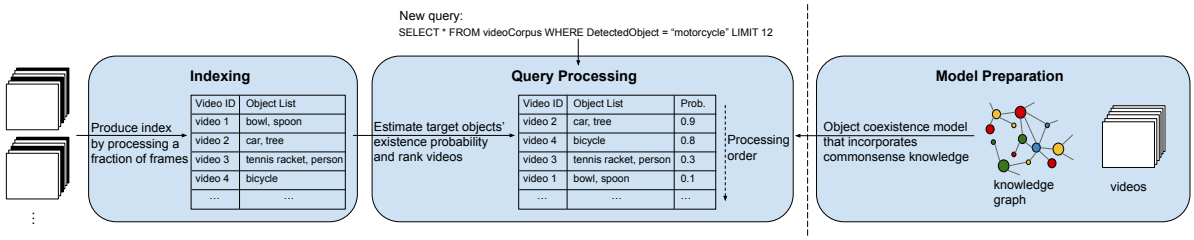

Figure 1: The three-stage architecture of PAINE system.

from the probability conditioned on each object $I_{(i,j)}$:

$$P(O|\mathcal{I}_i) \approx 1 - (1 - P(O|I_{(i,1)}))(1 - P(O|I_{(i,2)})) \cdots (1 - P(O|I_{(i,m_i)})) \quad (2)$$

Here the probability $P(O|I_{(i,j)})$ is estimated as Equation (1).

When there are multiple target objects in the predicate, PAINE roughly estimates $P(O|I_{(i,j)})$ by the product of the probability of each target object conditioned on $I_{(i,j)}$, and reuses Equation (2).

2.3.2 *Video-Derived Model.*

The system can compute object observation probabilities from a set of lists, each of which contains objects observed in the same video. These lists can be obtained by running a detector over a set of videos, e.g., videos from the same source as the queried videos. Objects frequently coexist in these videos tend to coexist in the queried videos. If such videos are available, this video-specific commonsense knowledge is likely to be more beneficial than the above knowledge graphs.

PAINE directly learns a neural network model based on observed object statistics derived from processing a collection of video. The input of this model is a set of observed objects and target objects concatenated together. Deduplicated observed objects are arranged in the time order (experimental evidence shows that object frequency does not help). PAINE keeps the outputs between 0 and 1 as the probability estimation indicating how likely this video contains target objects. This model consists of the pre-trained uncased BERT (Bidirectional Encoder Representations from Transformers) base model [4] and a regression layer to be fine-tuned by training data. It takes hours to fine-tune the model offline, which is subsequently utilized for multiple queries.

This model is trained with all the potential single target objects since the desired object is not pre-defined in this stage. For each object list in the observed object statistics, PAINE sets each item in this list as the target object in turn, and treats the remaining items as the observed objects, making the training data with label 1; it randomly assigns a new object as the target object and uses the whole object list as the observed list, making the training data with label 0. When more observed object statistics are gained from the *Query Processing* stage, PAINE further updates this model.

3 DEMO SCENARIO

In this section, we demonstrate a prototype system of PAINE on YouTube videos in diverse domains.

Step 1: The user will be presented with a system that has indexed 200 YouTube video clips. They are 60-second video clips randomly sampled from the YouTube-8M Segments dataset [1]. Considering

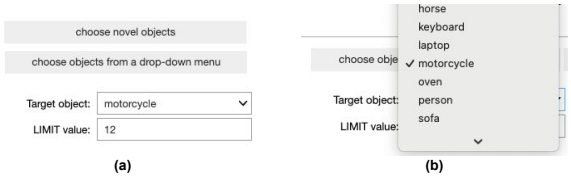

Figure 2: User Interface of PAINE

the demo time, we choose this size and length of the video set, but our system can work for more and longer videos. The index was built by applying YOLO9000 [11] at the rate of one frame per second. This system has also learned a video-derived model from the YouTube-8M Segments dataset other than the above 200 video clips. The user will be able to interact with the system by specifying the target object and the LIMIT value (Figure 2(a)). She can click the button "choose objects from a drop-down menu" and choose from a pre-selected list of objects (Figure 2(b) only shows a subset of the target object options). She can also click "choose novel objects" and enter novel object names. As shown in Figure 2, the user has chosen parameters that yield the SQL query example in Section 1.

Step 2: She can run the query and compare the query results and the query runtime of different optimization methods. This query can be run in two modes: the SCAN mode and the PAINE commonsense indexing mode. The SCAN method leverages the same index as PAINE. It first returns videos where the target object happened to be observed directly and is listed in the index. If these videos are not enough to answer the query, SCAN will then process videos sequentially. The PAINE commonsense indexing mode adopts the video-derived model. We set a two-hour timeout for the query processing. The demo will have precomputed results for target objects so the user does not have to actually wait for two full hours.

She will see a summary table (the first two rows in Figure 3) showing the query results and the execution time of these two modes. In the SCAN mode, this motorcycle query cannot be finished within two hours; only eight videos can be returned to the user. Our system PAINE runs dramatically faster than SCAN. It only takes 20 minutes to select high-quality videos that contain motorcycles.

Step 3: The user can decide to dig into the internal workings to figure out what factors affect the query runtime. She will be able to examine the output of line 6 in Algorithm 1 (videos that contain objects that were directly observed and so are definitely in the index), the output of line 10 in Algorithm 1 (videos that were not directly observed but found by examining frames during query processing),

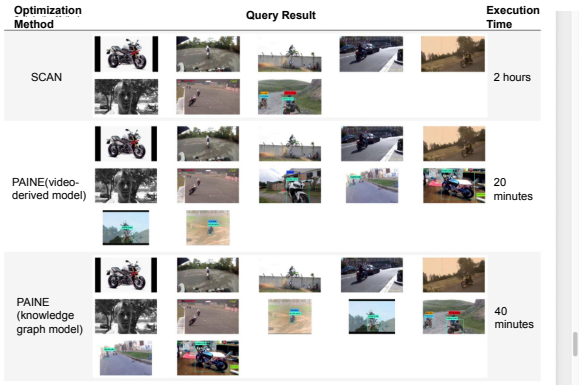

Figure 3: Summary table to compare SCAN and PAINE for the motorcycle selection query

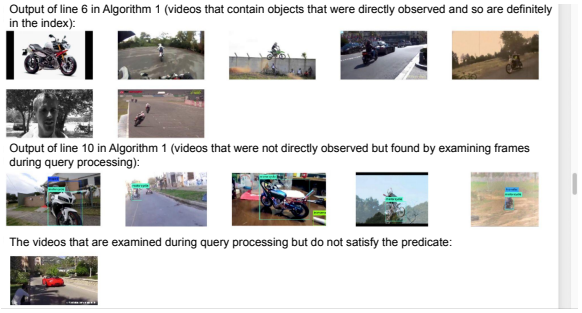

Figure 4: Internal working of PAINE with the video-derived model for the motorcycle selection query

and the videos that are examined during query processing but do not satisfy the predicate. These three categories are denoted as DIRECTLY OBSERVED, DISCOVERED, and IRRELEVANT videos. Figure 4 shows the internal working of the PAINE mode as an example. There are seven DIRECTLY OBSERVED videos and five DISCOVERED videos. As shown in the bottom part of Figure 4, the commonsense knowledge can instruct the query engine to process only one IRRELEVANT video, explaining the fast processing in step 2. In contrast, in the SCAN mode, there is only one DISCOVERED video and tons of IRRELEVANT videos. It would take much longer than two hours to finish this query in the SCAN mode.

Step 4: The user is also allowed to try another commonsense knowledge model in PAINE, the knowledge graph model. She can run the selection query with this model and will see a summary table (Figure 3) comparing it with other optimization methods. It takes 40 minutes to complete the process, which is slower than the video-derived model but much faster than the SCAN method. She can also examine the internal working of this method as in step 3. Same as Figure 4, there are seven DIRECTLY OBSERVED videos and five DISCOVERED videos, but more IRRELEVANT videos are processed.

Step 5: The user can go back to step 1 to try other video selection queries. For example, she can click the button "choose novel objects" and enter "chair" as the target object and 11 as the LIMIT value.

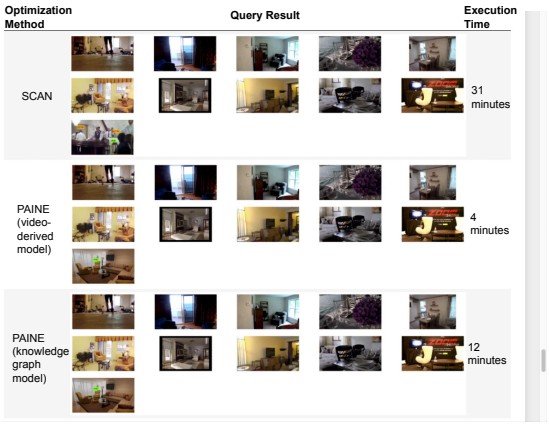

Figure 5: Summary table for the chair selection query

After repeating the above steps, she will see a new summary table as shown in Figure 5.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of National Science Foundation convergence accelerator award A6940.

REFERENCES

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675* (2016).

[2] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. 2019. Physical representation-based predicate optimization for a visual analytics database. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1466–1477.

[3] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. 2021. TASM: A tile-based storage manager for video analytics. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1775–1786.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046* (2018).

[6] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529* (2017).

[7] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1725–1732.

[8] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: a large video database for human motion recognition. In *2011 International conference on computer vision*. IEEE, 2556–2563.

[9] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*. 1493–1508.

[10] Thomas Marcoux, Nitin Agarwal, Recep Erol, Adewale Obadimu, and Muhammad Nihal Hussain. 2021. Analyzing cyber influence campaigns on YouTube using YouTubeTracker. In *Big Data and Social Media Analytics*. Springer, 101–111.

[11] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

[12] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

[13] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022).