

Rekall: Specifying Video Events using Compositions of Spatiotemporal Labels

Daniel Y. Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong,
Avanika Narayan, Maneesh Agrawala, Christopher Ré, Kayvon Fatahalian
Stanford University

{danfu, wcrichto, james.hong, xinwei.yao, haotianz, anhlt92, avanika, maneesh,
chrismre, kayvonf}@cs.stanford.edu

ABSTRACT

Many real-world video analysis applications require the ability to identify domain-specific events in video, such as interviews and commercials in TV news broadcasts, or action sequences in film. Unfortunately, pre-trained models to detect all the events of interest in video may not exist, and training new models from scratch can be costly and labor-intensive. In this paper, we explore the utility of specifying new events in video in a more traditional manner: by writing queries that compose outputs of existing, pre-trained models. To write these queries, we have developed REKALL, a library that exposes a data model and programming model for compositional video event specification. REKALL represents video annotations from different sources (object detectors, transcripts, etc.) as spatiotemporal labels associated with continuous volumes of spacetime in a video, and provides operators for composing labels into queries that model new video events. We demonstrate the use of REKALL in analyzing video from cable TV news broadcasts, films, static-camera vehicular video streams, and commercial autonomous vehicle logs. In these efforts, domain experts were able to quickly (in a few hours to a day) author queries that enabled the accurate detection of new events (on par with, and in some cases much more accurate than, learned approaches) and to rapidly retrieve video clips for human-in-the-loop tasks such as video content curation and training data curation. Finally, in a user study, novice users of REKALL were able to author queries to retrieve new events in video given just one hour of query development time.

1. INTRODUCTION

Modern machine learning techniques can robustly annotate large video collections with basic information about their audiovisual contents (e.g., face bounding boxes, people/object locations, time-aligned transcripts). However, many real-world video applications require exploring a more diverse set of events in video. For example, our recent efforts to analyze cable TV news broadcasts required models to detect interview segments and commercials. A film production team may wish to quickly find common segments such as action sequences to put into a movie trailer. An autonomous vehicle development team might wish to mine video collections for events like traffic light changes or obstructed left turns to debug the car’s prediction and control systems. A machine learning engineer developing a new model for video analysis may search for particular scenarios to bootstrap model development or focus labeler effort.

Unfortunately, pre-trained models to detect these domain-specific events often do not exist, given the large number and diversity of potential events of interest. Training models for new events can be difficult and expensive, due to the large cost of labeling a training set from scratch, and the computation time and human skill

required to then train an accurate model. We seek to enable more agile video analysis workflows where an analyst, faced with a video dataset and an idea for a new event of interest (but only a small number of labeled examples, if any), can quickly author an initial model for the event, immediately inspect the model’s results, and then iteratively refine the model to meet the accuracy needs of the end task.

To enable these agile, human-in-the-loop video analysis workflows, we propose taking a more traditional approach: *specifying novel events in video as queries that programmatically compose the outputs of existing, pre-trained models*. Since heuristic composition does not require additional model training and is cheap to evaluate, analysts can immediately inspect query results as they iteratively refine queries to overcome challenges such as modeling complex event structure and dealing with imperfect source video annotations (missed object detections, misaligned transcripts, etc.).

To explore the utility of a query-based approach for detecting novel events of interest in video, we introduce REKALL, a library that exposes a data model and programming model for *compositional video event specification*. REKALL adapts ideas from multi-media databases [4, 15, 23, 31, 32, 36, 40] to the modern video analysis landscape, where using the outputs of modern machine learning techniques allows for more powerful and expressive queries, and adapts ideas from complex event processing systems for temporal data streams [8, 16, 25] to the spatiotemporal domain of video.

The primary technical challenge in building REKALL was defining the appropriate abstractions and compositional primitives for users to write queries over video. In order to compose video annotations from multiple data sources that may be sampled at different temporal resolutions (e.g., a car detection on a single frame from a deep neural network, the duration of a word over half a second in a transcript), REKALL’s data model adopts a unified representation of multi-modal video annotations, the *spatiotemporal label*, that is associated with a continuous volume of spacetime in a video. REKALL’s programming model uses hierarchical composition of these labels to express complex event structure and define increasingly higher-level video events.

We demonstrate the effectiveness of compositional video event specification by implementing REKALL queries for a range of video analysis tasks drawn from four application domains: media bias studies of cable TV news broadcasts, cinematography studies of Hollywood films, analysis of static-camera vehicular video streams, and data mining autonomous vehicle logs. In these efforts, REKALL queries developed by domain experts with little prior REKALL experience achieved accuracies on par with, and sometimes significantly better than, those of learning-based approaches (6.1 F1 points more accurate on average, and up to 26.1 F1 points more accurate for one task). REKALL queries also served as a key video data re-

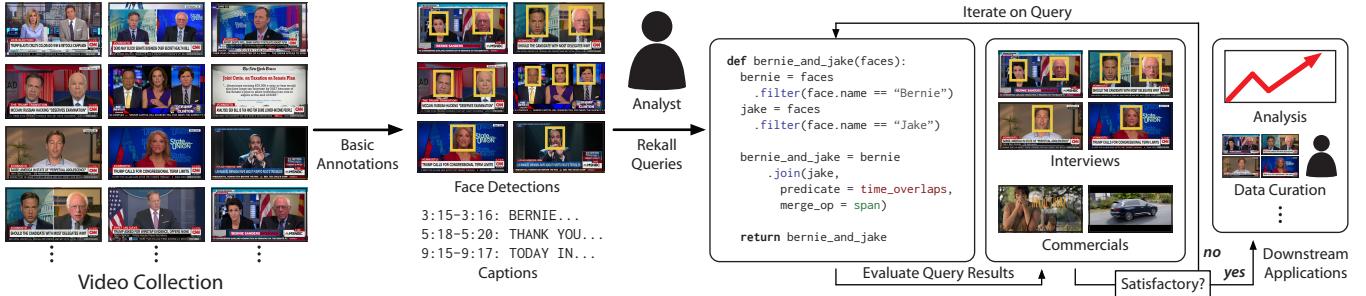


Figure 1: Overview of a compositional video event specification workflow. An analyst pre-processes a video collection to extract basic annotations about its contents (e.g., face detections from an off-the-shelf deep neural network and audio-aligned transcripts). The analyst then writes and iteratively refines REKALL queries that compose these annotations to specify new events of interest, until query outputs are satisfactory for use by downstream analysis applications.

trieval component of human-in-the-loop exploratory video analysis tasks.

Since our goal is to enable analysts to quickly retrieve novel events in video, we also evaluate how well users are able to formulate REKALL queries for new events in a user study. We taught participants how to use REKALL with a one-hour tutorial, and then gave them one hour to write a REKALL query to detect empty parking spaces given the outputs of an off-the-shelf object detector. Users with sufficient programming experience were able to write REKALL queries to express complex spatiotemporal event structures; these queries, after some manual tuning (changing a single line of code) by an expert REKALL user to account for failures in the object detector, achieved near-perfect accuracies (average precision scores above 94).

To summarize, in this paper we make the following contributions:

- We propose compositional video event specification as a human-in-the-loop approach to rapidly detecting novel events of interest in video.
- We introduce REKALL, a library that exposes a data model and programming model for compositional video event specification by adapting ideas from multi-media databases and complex event processing over temporal data streams to the modern video analysis landscape.
- We demonstrate the effectiveness of REKALL through analysis tasks across four application domains, where domain experts were able to quickly author REKALL queries to accurately detect new events (on average 6.1 F1 points more accurate, and up to 26.1 F1 points more accurate, than learned approaches) and support human-in-the-loop video retrieval workflows.
- We evaluate how well novice users of REKALL are able to detect a novel event in video given a one-hour tutorial and one hour of query development time (average precision scores above 94 after tuning by an expert).

The rest of this paper is organized as follows: Section 2 introduces an interview detection running example. Section 3 and 4 use the running example to introduce REKALL’s data model and programming model. Section 5 introduces our application domains and analysis tasks, and in Section 6 we evaluate the accuracy of the REKALL queries used to solve these tasks and evaluate the usability of REKALL for video event specification. Finally, we conclude with related work and discussion in Sections 7 and 8.

Some supplemental videos can be found at <http://www.danfu.org/projects/rekall-vldb2020/>.

2. AN ANALYSIS EXAMPLE

To better understand the thought process underlying our video analysis tasks, consider a situation where an analyst, seeking to understand sources of bias in TV political coverage, wishes to tabulate the total time spent interviewing a political candidate in a large collection of TV news video. Performing this analysis requires identifying video segments that contain interviews of the candidate. Since extracting TV news interviews is a unique task, we assume a pre-trained computer vision model is not available to the analyst. However, it is reasonable to expect an analyst does have access to widely available tools for detecting and identifying faces in the video, and to the video’s time-aligned text transcripts.

Common knowledge of TV news broadcasts suggests that interview segments tend to feature shots containing faces of the candidate and the show’s host framed together, interleaved with headshots of just the candidate. Therefore, a first try at an interview detector query might attempt to find segments featuring this temporal pattern of face detections. Refinements to this initial query might permit the desired pattern to contain brief periods where neither individual is on screen (e.g., display of B-roll footage for the candidate to comment on), or require parts of the sequence to align with utterances of the candidate’s name in the transcript or common phrases like “welcome” and “thank you for being here.” As illustrated in Figure 1, arriving at an accurate query for a dataset often requires multiple iterations of the analyst reviewing query results and adding additional heuristics as necessary until a desired level of accuracy is achieved.

Even in this simple example, a number of challenges emerge. Annotations used as query inputs may be of different modalities and sampled at different temporal rates (e.g., face detections are computed per frame, transcript text is sub-second aligned). Queries must be robust to noise in source annotations (e.g., missed face detections, partially misaligned transcript data). Last, to be sufficiently expressive to describe a range of events, the system must provide a rich set of composition operators to describe temporal and (although not required in this example) spatial relationships between annotations.

The following sections describe REKALL’s data model – its representation of multi-modal video annotation inputs – and its programming model, the operations available to queries for defining new video events in terms of these inputs.

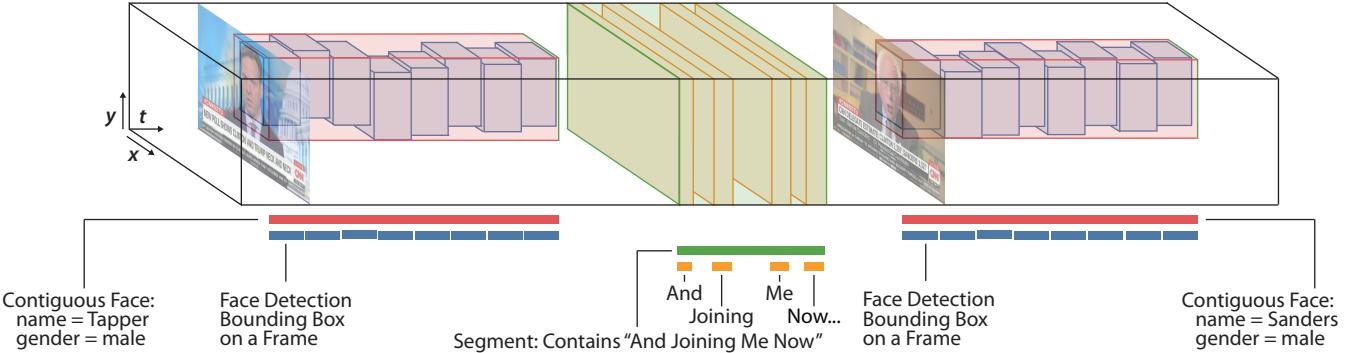


Figure 2: REKALL represents all video annotations, both basic annotations from computer vision models and annotations of more complex events, as labels associated with spatiotemporal intervals in the domain of a video. REKALL’s labels can be nested. We illustrate two labels representing video segments where a face is continuously on screen (red) that contain labels corresponding to per-frame face detections (blue), and one caption segment (green) that contains labels for individual words (orange).

3. SPATIOTEMPORAL LABELS

To facilitate queries that combine information from a video sampled at different rates and originating from different source modalities, REKALL adopts a unified representation for all data: a *spatiotemporal label* (or *label*). Similar to how temporal databases associate records with an interval of time designating their insertion to and deletion from the database [26], each REKALL label is associated with a continuous, axis-aligned interval of spacetime that locates the label in a video (a label is the equivalent of a database record in REKALL’s data model). For example, a face detected in a frame two minutes and ten seconds into a 30 fps video with bounding box co-ordinates $\{x_1: 0.2, x_2: 0.4, y_1: 0.2, y_2: 0.8\}$ (relative to the size of the frame) yields a label whose interval spans this box in space and the range $\{t_1: 2:10.0, t_2: 2:10.333\}$ in time. REKALL labels also optionally include metadata. For example, a face detection label might include the name of the detected individual:

```
face = Label(  
    Interval=(t1: 2:10.0, t2: 2:10.333,  
              x1: 0.2, x2: 0.4,  
              y1: 0.2, y2: 0.8),  
    Metadata={ identity: Bernie Sanders }  
)
```

Figure 2 illustrates examples of labels generated for the TV news interview task introduced in Section 2. Face detection performed on each frame yields labels (one per detected face, blue boxes) that span one frame of time (with the name of the individual as metadata). The results of time-aligning the video’s transcript yields a label per word that extends for the length of the utterance (with the word as metadata, yellow boxes). Although words in a transcript are inherently temporal (and not spatial), they can be lifted to the full spatiotemporal domain by assigning them the entire space of the frame. Most examples in this paper use a 3D (X,Y,T) video domain, although REKALL also supports intervals with additional spatial dimensions (e.g., labels in a LIDAR point cloud video exist in a 4D domain).

To echo the hierarchical nature of information derived from a video, labels in REKALL queries can be hierarchical (a label’s metadata can be a list of labels.) For example, a set of temporally continuous face detections of the same individual at the same location on screen might be grouped into a single label representing a segment where the individual is on screen. The red boxes in Figure 2 indicate segments where anchor Jack Tapper or guest Bernie Sanders are on screen. The figure also shows a label corresponding to the phrase “And joining me now” that contains labels for the

constituent words (green box). Many of the queries described in the subsequent sections construct multi-level hierarchies of labels. For example, in TV and film videos, frames can be organized into shots, and shots can be grouped into scenes (or news segments).

4. COMPOSING LABELS

REKALL queries define how to compose existing labels into new labels that correspond to instances of new events in a video. In this section we describe the label composition primitives available to REKALL queries. To aid description, Figure 3 provides code for the TV news interview detection task from Section 2, which will be used as the running example throughout this section.

4.1 Label Sets

REKALL uses sets of labels to represent all occurrences of an event in a video (a label set is equivalent to a database relation). A REKALL query consists of operations that produce and consume sets of labels (all REKALL operations are closed on sets of labels). For example, Line 1 of Figure 3 constructs an initial label set from a database table containing all face detections from a video. The variable `faces` is a set containing one label for each detected face. The result of the query is the set `interviews`, which contains one label corresponding to each interview segment in the video.

REKALL provides standard data-parallel operations `map`, `filter`, and `group_by` to manipulate label sets. For example, lines 3 and 11 filter faces according to the person’s name (pre-computed using off-the-shelf identity recognition tools [1] and stored as label metadata) to produce label sets containing only detections of Jake Tapper (`tapper`) and Bernie Sanders (`sanders`).

Figure 4 illustrates the behavior of various REKALL label set operations. `map` can be used to manipulate the metadata or the spatiotemporal interval associated with labels (e.g., shrink the interval as shown in the figure). `group_by` is one mechanism by which REKALL queries construct nested labels. For example, Figure 4, bottom-left shows the use of `group_by` to reorganize a set of four face detection labels into a set of two labels that each contain a set of labels corresponding to faces in the same frame.

4.2 Coalesce: Recursive Label Merging

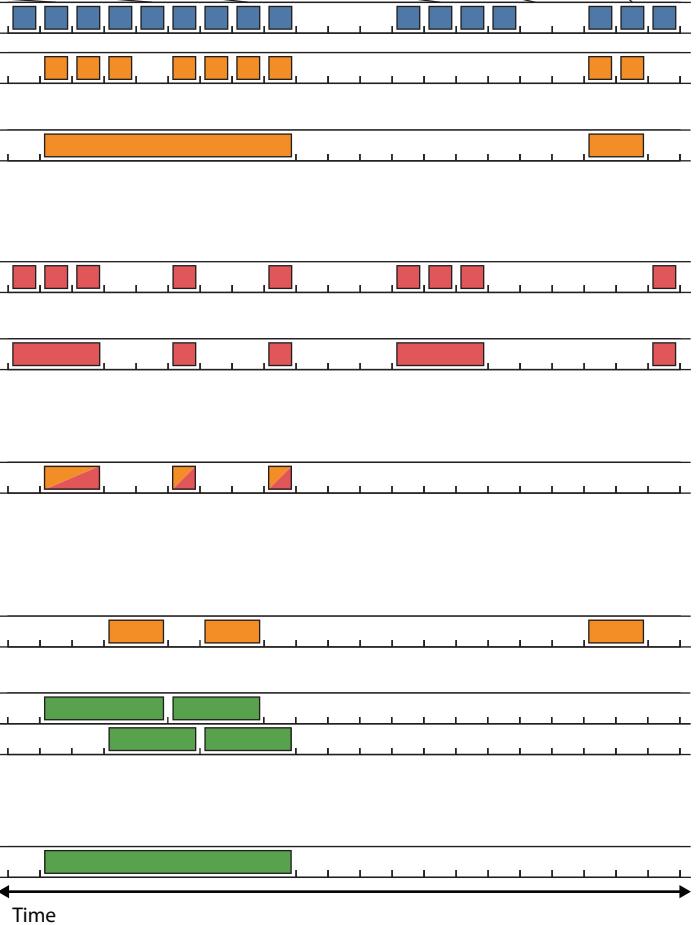
Many video analysis tasks involve reasoning about sequences of labels or about spatially adjacent labels. For example, in the TV interviews query, it is preferable to reason about continuous segments of time where a person is on screen (“a segment containing Bernie



```

1 faces = rekall.ingest(database.table("faces"), 3D)
2
3 sanders = faces
4     .filter(lambda face: face.name == "Bernie Sanders")
5
6 sanders_segs = sanders
7     .coalesce(
8         predicate = time_gap < 30 seconds,
9         merge = time_span)
10
11 tapper = faces
12     .filter(lambda face: face.name == "Jake Tapper")
13
14 tapper_segs = tapper
15     .coalesce(
16         predicate = time_gap < 30 seconds,
17         merge = time_span)
18
19 sanders_and_tapper_segs = sanders_segs
20     .join(
21         tapper_segs,
22         predicate = time_overlaps,
23         merge = time_intersection)
24
25 sanders_alone_segs = sanders_segs
26     .minus(sanders_and_tapper_segs)
27
28 interview_segs = sanders_and_tapper_segs
29     .join(
30         sanders_alone_segs,
31         predicate = before or after,
32         merge = time_span)
33
34 interviews = interview_segs.coalesce()

```



■ Face Detections ■ Bernie Sanders Detections ■ Jake Tapper Detections ■ Bernie Sanders and Jake Tapper ■ Interviews

Figure 3: Left: A REKALL query for detecting interviews of Bernie Sanders by Jack Tapper. Right: A visual depiction of the intermediate label sets produced over the course of execution. Blue: face detections; Orange: Bernie Sanders; Red: Jake Tapper; Orange and Red: Bernie Sanders and Jake Tapper; Green: interview segments.

Sanders on screen, followed by one with Jake Tapper”), rather than individual frame face detections.

REKALL’s `coalesce` operator serves to merge an unbounded number of fine-grained labels in close proximity in time or space into new labels that correspond to higher-level concepts. `coalesce` is parameterized by a query-specified *label merge predicate*, which determines whether a pair of labels should be merged, and a *label merge function* that specifies how to create a new label as a result of a merge. `coalesce` recursively merges labels in its input set (using the merge function), until no further merges are possible (as determined by the merge predicate).

Lines 6 and 14 in Figure 3 demonstrate use of `coalesce` to merge label sets of per-frame face detections into label sets corresponding to sequences of time when Bernie Sanders and Jake Tapper are on screen. In this example, the query uses a merge predicate that merges all input labels that lie within 30 seconds of each other (not just temporally adjacent labels) and the merge function combines

the two labels to create a new label whose spatiotemporal interval spans the union of the intervals of the two inputs. As a result, the resulting labels, which correspond to video segments showing the interviewer or interviewee on screen, may contain brief cuts away from the individual.

`coalesce` serves a similar role as “repeat” operators in prior multimedia database systems [4, 15, 23, 31, 32, 36, 40], or as a regex Kleene-star operator on point events in event processing systems [8, 16]. However, it is a general mechanism that provides queries the flexibility to build increasingly higher levels of abstraction in custom ways. For example, it is common to use `coalesce` with a spatiotemporal proximity predicate to build labels that correspond to “tracks” of an object (Figure 4, bottom-center) or to smooth over noise in fine-grained labels (e.g., flicker in an object detector). We document a variety of use cases of `coalesce` in Section 5.

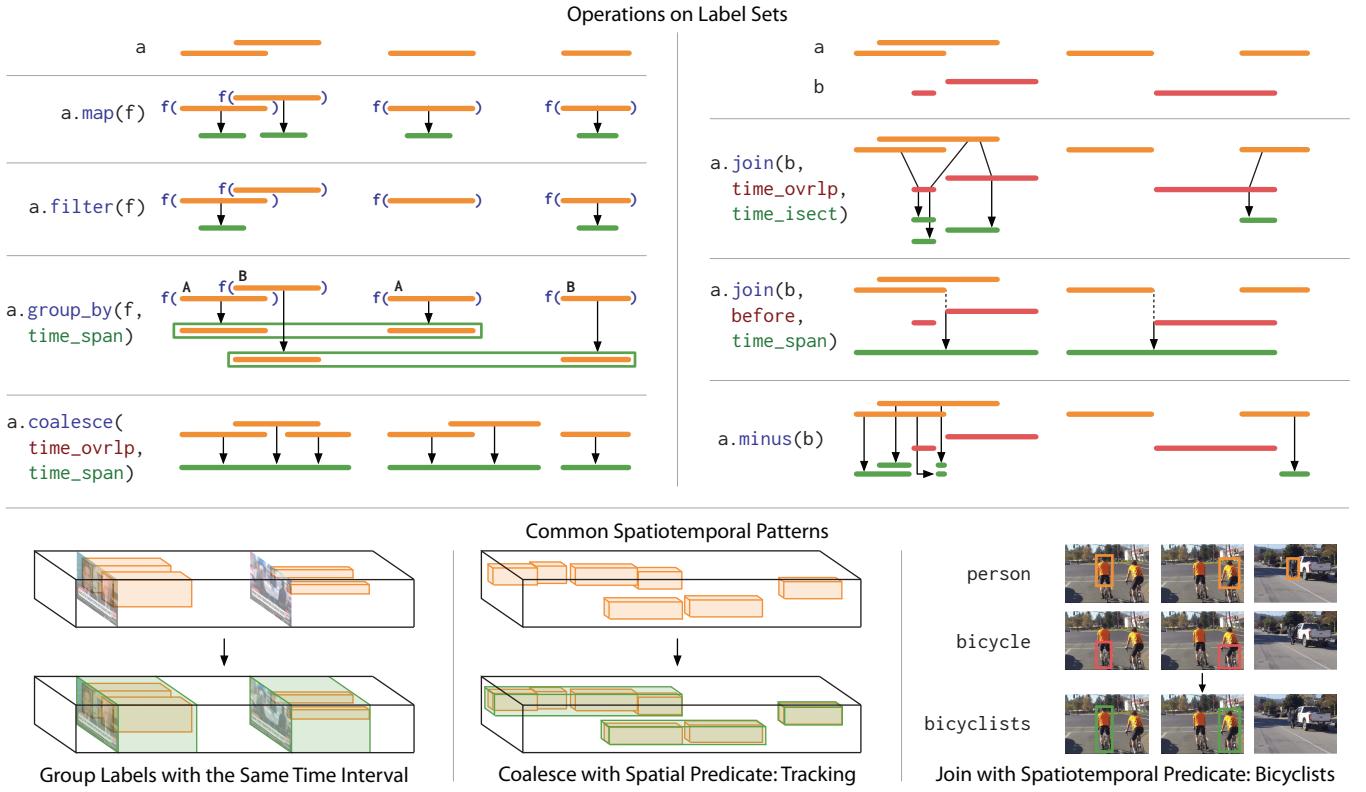


Figure 4: Semantics of REKALL operations on label sets. The functions shown are a sample of REKALL’s complete library of operators, intended to provide visual intuition for how a few relevant operators work. Top half: label set operations, depicted on sets of one-dimensional temporal labels. Bottom half: depictions of common spatiotemporal operations on multi-dimensional labels.

4.3 Joins

Video analysis tasks often involve reasoning about multiple concurrent label streams; REKALL provides standard join operators to combine multiple streams together and construct new labels from spatiotemporal relationships between existing labels. REKALL’s inner `join` is parameterized by a join predicate and a label merge function that specifies how to merge matching pairs of labels (like `coalesce`). For example, line 19 in Figure 3 uses `join` with a temporal overlap predicate (`time_overlaps`) and a temporal intersection merge function (`time_intersection`) to generate a label set (`sanders_and_tapper_segs`) corresponding to times when both Sanders and Tapper are on screen. Line 28 in the pseudocode uses a different join predicate (`before or after`) and merge function (`time_span`) to construct labels for segments where Sanders is on screen directly before or directly after a segment containing both Sanders and Tapper.

As shown at right in Figure 4, REKALL provides a standard library of common spatial and temporal predicates such as the Allen interval operations [5] and their 2D spatial analogues [9]. The bottom-right of the Figure illustrates use of a join with a predicate (“above”) to construct labels for bicyclists from label sets of bicycle and person detections.

REKALL’s `minus` operation is an anti-semi-join (similar to Trill’s `WhereNotExists` [8]) that takes two label sets and removes intervals associated with labels in the second set from the intervals of labels in the first. For example, Line 10 in Figure 3 uses `minus` to construct labels for segments when Bernie Sanders is on screen alone. Like `join`, `minus` is parameterized by a predicate that determines which labels are matched by the operation. The empty parking space detection query shown in Figure 7 illustrates use of `minus`.

with a spatiotemporal predicate that only matches labels when their interval intersection to union ratio (IOU) exceeds a manually-set threshold.

5. APPLICATIONS

We have used REKALL to write queries needed for video analysis tasks in several domains: media bias studies of TV news broadcasts, analysis of cinematography trends in feature length films, event detection in static-camera vehicular video streams, and data mining the contents of autonomous vehicle logs. In many cases, these queries have been used to automatically label large video collections for statistical analysis; in other cases, REKALL queries have also proven to be a valuable mechanism for retrieving video clips of interest in scenarios that involve human-in-the-loop analysis tasks.

The remainder of this section provides further detail on how REKALL’s programming model was used to author queries used in these application domains. Table 1 enumerates several tasks from REKALL’s deployments, and includes the basic annotations used as input to REKALL queries that perform these tasks. Code listings for SHOT SCALE and PARKING are provided in this section, and code listings for additional queries are provided in the Appendix.

5.1 Analysis of Cable TV News Media

We have used REKALL queries as part of an ongoing study of representation and bias in over 200,000 hours of U.S. cable TV news (CNN, MSNBC, FOX) between 2010 and 2018. This effort seeks to analyze differences in screen time afforded to individuals of different demographic groups, and asks questions such as “Did Donald Trump or Hillary Clinton get more interview screen time in

Task	Application(s)	Data Sources	Description
Commercial Detection (COMMERCIAL)	TV News	Histograms, transcripts	Detect all commercial segments
Interview Detection (INTERVIEW)	TV News	Face detections	Detect all interviews with a particular guest (e.g., Bernie Sanders)
Shot Transition Detection (SHOT DETECT)	Film	Histograms, face detections	Detect every shot transition
Shot Scale Classification (SHOT SCALE)	Film	Face detections, pose estimations	Classify the scale of each shot
Conversation Detection (CONVERSATION)	Film	Face detections, face embeddings	Detect all conversations
Film Idiom Mining (FILM IDIOM)	Film	Face detections, transcripts, histograms	Detect various film idioms – reaction shots, action sequences, establishing shots, etc.
Empty Parking Space Detection (PARKING)	Static-Camera Feeds	Object detections	Detect all empty parking spots
AV Log Mining (AV)	AV	Object detections from cameras, LIDAR	Detect various rare events from autonomous vehicle logs
Upstream Model Debugging (DEBUGGING)	TV News, Film, AV, Static-Camera Feeds	Model outputs	Detect errors in model outputs

Table 1: Nine representative tasks from REKALL deployments for analysis of a large collection of cable TV News, cinematographic studies of Hollywood films, analysis of static-camera vehicular video feeds, and data mining commercial autonomous vehicle logs.

the months before the 2016 election?” or “How much more screen time is given to male presenting vs. female presenting hosts?” To answer these questions, screen time aggregations needed to be scoped to specific video contexts, such as interview segments, and needed to exclude commercials. Thus, a key challenge involved developing queries for accurately detecting commercial segments (COMMERCIAL) and segments featuring interviews with specific political candidates (INTERVIEW).

A simplified version of the interview detection algorithm was described in Section 4; in practice, we extend the algorithm to find interviews between *any* individual known to be a cable TV news host and a specified guest (we used Jake Tapper for expositional simplicity; modifying the query in Figure 3 to find candidate interviews with any host is a one-line change). In the TV news dataset, commercial segments often begin and end with short sequences of black frames and often have mixed-case or missing transcripts (news broadcasts typically feature upper-case caption text). The COMMERCIAL query exploits these dataset-specific signals to identify commercial segments. More details can be found in the Appendix.

5.2 Film Cinematography Studies

We have analyzed a collection of 589 feature-length films spanning 1915-2016 to explore questions about cinematographic techniques, and how their use has changed over time (e.g., “*How has the pace or framing of films changed over the past century?*”, or “*How much screen time do conversations take in films?*” [7, 10–14]). Representative queries are depicted in Figure 5, and include shot transition detection (SHOT DETECT, partitioning a video into segments of continuously recorded footage), classifying shots by the relative size of the actors to the size of the frame (SHOT SCALE), and conversation detection (CONVERSATION). Our film analyses also required queries for retrieving segments exhibiting common film cinematography idioms such as action sequences or wide-angle scene “establishing shots” for video clip content curation (FILM IDIOM).

As one example of a query used in this effort, Figure 6 provides code for SHOT SCALE, which classifies each shot in a video as “long”, “medium”, or “close up” based on the size of actors on screen. Inputs to this query include a label set for all film shots (shots), as well as label sets for per-frame face detections faces and actor body poses (poses). Each shot contains multiple frames, each of which contains zero or more actors. The challenge of this query is to use the relative sizes of each actor in a frame to estimate the scale for the frame, and then use per-frame scale estimates to estimate the scale of the shot. The REKALL query echoes this nested

structure using hierarchical labels to estimate the scale of a shot.

The query first estimates the scale based on each face detection or pose detection in a frame. Lines 10 and 12 estimate the scale based on the relative size of face bounding boxes or pose skeletons, using `frame_scale_face` or `frame_scale_pose`, respectively. The query then aggregates these estimates into a single label set for each frame, retaining the largest estimate (`take_largest`) from all detected faces and poses (lines 15–18). Finally, the query identifies the frames contained within each shot and classifies the shot’s scale as the mode of the scales computed for each constituent frame (lines 20–26).

We provide details about the other cinematography queries in the Appendix. These queries use rapid changes in video frame pixel color histograms and face bounding box locations to detect shot boundaries, identify patterns of shots where the same two individuals appear over an extended period of time as a signal for conversations, and use patterns in length or scale of consecutive shots to identify common film idioms.

Human-in-the-loop Movie Data Exploration. In addition to conducting film analyses, we have also used REKALL queries as a tool for exploring films and curating content needed for video editing tasks like making video supercuts or movie trailers [6, 54]. These video editing tasks require finding clips in a film that embody common film idioms such as action shots, “hero shots” of the main characters, or wide-angle establishing shots.

We authored REKALL queries for several different types of film idioms, and provided query results to a video editor who selected final clips to use in the movie trailer. Query recall in this task was more important than precision, since the editor could quickly result sets to select a small number of desirable clips. We demonstrated this workflow on *Star Wars: Episode III - Revenge of the Sith*. Once queries were written, the entire mining and trailer construction process took less than four hours. The full list of film idioms we mined for this effort can be found in the Appendix. The movie trailer, a selection of supercuts, and examples of other film idioms can be found at <http://www.danfu.org/projects/rekall-vldb2020/>.

5.3 Static-Camera Vehicular Video Streams

Inspired by recent vision applications in the wild [19], PARKING detects the time periods where parking spots are free in a fixed-camera video feed of a parking lot. The query, whose code is given in Figure 7, uses only labels produced by an off-the-shelf object detector run on the video stream, and is based on two simple heuristics: a parking spot is a spatial location where a car is stationary for a long period of time, and an empty parking spot is a parking spot



Figure 5: Examples of film idioms extracted from *Star Wars: Episode III - Revenge of the Sith* using REKALL queries.



```

1 faces = rekall.ingest(database.table("faces"), 3D)
2 poses = rekall.ingest(database.table("poses"), 3D)
3 shots = rekall.ingest(database.table("shots"), 1D)

4
5 faces_per_frame = faces
6   .group_by(λ obj: (obj["t1"], obj["t2"]), span)
7 poses_per_frame = poses
8   .group_by(λ obj: (obj["t1"], obj["t2"]), span)

9
10 frame_scales_face = faces_per_frame
11   .map(frame_scale_face)
12 frame_scales_pose = poses_per_frame
13   .map(frame_scale_pose)

14
15 frame_scales = frame_scales_face
16   .union(frame_scales_pose)
17   .group_by(λ frame: (frame["t1"], frame["t2"]), span)
18   .map(λ frame: take_largest(frame.nested))

19
20 shot_scales = frame_scales
21   .join(
22     shots,
23     predicate = time_overlaps,
24     merge = time_span)
25   .group_by(λ shot: (shot["t1"], shot["t2"]), span)
26   .map(λ shot: mode(shot.nested))

```

Figure 6: A query for classifying the scale of a cinematic shot (as long, medium, or close up) based on the size of faces and human pose estimates detected in frames. The query uses nested labels to group per-detection estimates of scale by frame, then pools per-frame results by shot to arrive at a final estimate.

without a car in it.

For simplicity, the query in Figure 7 assumes that all parking spaces are taken at the start of the video (this assumption can be relaxed by extending the query to identify regions that are occupied by cars for a significant period of time at any point in the video). The query extends the car detections at the start of the video to the entire video to construct a label set of parking spots (lines 3-5), and then subtracts out car and truck detections (lines 7-12). The predicate `iou` (intersection-over-union) provided to the `minus` operator ensures that the operation only removes times when the parking spot is completely filled (and not when vehicles partially overlap in pixel space when passing by). The behavior of the spatiotemporal



```

1 objects = rekall.ingest(database.table("objects"), 3D)
2
3 parking_spots = objects
4   .filter(λ obj: obj["t1"] == 0 and obj.class == "car")
5   .map(λ spot: spot with "t1" = 0 and "t2" = video_end)

6
7 vehicles = objects
8   .filter(λ obj: obj.class in ["car", "truck"])

9
10 empty_spot_candidates = parking_spots
11   .minus(vehicles,
12     predicate = λ spot, car: iou(spot, car) > 0.25)

13
14 empty_parking_spots = empty_spot_candidates
15   .coalesce(
16     predicate = λ spot, car: iou(spot, car) == 1,
17     merge = span)
18   .filter(λ spot: spot["t2"] - spot["t1"] > 60 * 4)

```

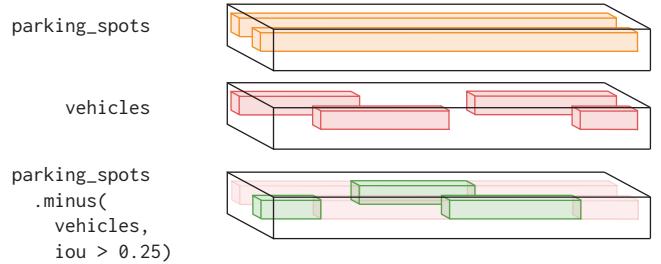


Figure 7: A query for detecting empty parking spaces using only object detection results in a fixed-camera feed. Potentially empty spots are obtained by subtracting the current frame's car intervals (red) from a set of intervals corresponding to all spots (orange). To account for errors in an object detector, a parking spot must be car free for a continuous period of time (4 minutes) to be counted as a “free spot”.

`minus` operation, and the surviving labels that correspond to empty parking spots (green intervals), is illustrated at the bottom of Fig-

ure 7. Finally, to avoid errors due to missed object detections, the query uses `coalesce` to construct consecutive empty spot detections, and removes detections that exist for less than four minutes (lines 14-18).

5.4 Mining Autonomous Vehicle Logs

REKALL queries are used at a major autonomous vehicle company to mine for rare, but potentially important, events in autonomous vehicle logs (AV). Queries are used to identify traffic light changes in quick succession, as well as turns by other vehicles near the autonomous vehicle. Images from the traffic light sequences are sent to human labelers for ground truth annotation, which may reveal correct detector behavior (a sequence of green-to-yellow-to-red transitions), major failures of the traffic light color detector (thus creating new labeled supervision for an active learning loop), or important rare situations to document such as a yellow flashing warning light. Images from vehicle turn sequences, on the other hand, are sent to labelers to label turn signals, and to validate the car’s prediction and control systems. As in the video editing case, since clips matching these queries are subsequently passed on to human labelers for review, REKALL queries serve as a filter that focuses human effort on examples that are most likely to be important.



Figure 8: Two examples using REKALL queries to debug computer vision models during our film cinematography studies. Top: Face detections “flickering” in and out of existence for a single frame at a time. Here, the face detector fails to detect Han Solo’s face in the middle frame, even though his face appears in the same location and the same place in the surrounding two frames. Bottom: The result of a query for frames with two women based on the results of an off-the-shelf gender classifier. Gender balance is so poor in fantasy movies that most of these examples are false positives. Male classifications are shown in blue; female classifications are shown in red. Harry Potter, Han Solo, Merry and Pippin have all been misclassified.

5.5 Model Debugging

In all the projects using REKALL, queries have been used to identify errors in the output of pre-trained models. A common example is shown in Figure 8, where a REKALL query is used to detect false negatives in a face detector from a lack of temporal coherence in its output (Han Solo’s face is not detected in the middle frame). Once users identify such errors, they often use `coalesce` to smooth over the errors to improve end results (PARKING smooths over errors in the object detector, for example, increasing accuracy by 19.8 AP points over a version of the query that does not). The `coalesce` operations in the INTERVIEW algorithm and aggregation steps in the SHOT SCALE algorithm played similar roles in those queries.

During our film study efforts, a query for detecting frames with at least two woman faces surfaced errors in the gender classifier, since scenes with multiple women in Hollywood films are rare due

to a bad gender imbalance. Most scenes retrieved by the query were false positives due to incorrect gender classification of faces. As a result, we subsequently replaced the face gender classifier with a better model. We note that our efforts searching for potentially anomalous patterns in trained model output is similar to recent work on improving models using model assertions [29].

6. EVALUATION

The goal of REKALL is to enable analysts to productively author queries that meet the requirements of a range of video analysis tasks. In this section, we evaluate REKALL in terms of the accuracy of the queries compared to learned approaches and the usability of REKALL by domain experts and novice users.

- In Section 6.1, we evaluate our ability to author high-accuracy REKALL queries by comparing query output to that of learned baselines (deep models trained on the same video data available to the programmer during query development). In five out of six representative tasks, REKALL queries were on par with, and in some cases significantly more accurate than, these learned baselines (6.1 F1 points more accurate on average across all tasks, and up to 26.1 F1 points more accurate in one case).
- Since developing a program that detects an event of interest in a video is more challenging than directly labeling instances of that event, we discuss the experiences of task domain experts authoring REKALL queries for real-world video analysis applications (Section 6.2).
- Finally, in a user study, we also evaluate how well novice REKALL users are able to author REKALL queries for a new detection task in one hour. Users with a sufficient functional programming background were able to author queries that achieved average precision (AP) scores above 94 after one-line tweaks by an expert user to account for errors in upstream object detectors.

6.1 Query Accuracy

We evaluate the accuracy of our REKALL queries by comparing query outputs to baseline learning-based approaches on six representative tasks for which high accuracy was required. For each task, we collect human-annotated ground truth labels, splitting labeled video into a *development set* that was made available to the REKALL programmer during query development, and a held-out *test set* used for evaluating the accuracy of REKALL queries and trained models. We train learning baselines using all human labels contained in the development set.

6.1.1 Classification Tasks

Five tasks (INTERVIEW, COMMERCIAL, CONVERSATION, SHOT DETECT, and SHOT SCALE) can be viewed as classification tasks. For these tasks, we compare REKALL query accuracy (F1 score) against those of image classification and action recognition networks trained on the development set (results in Table 2-top). These models classify whether a video frame or a short video segment (for the action recognition baseline) contains the event of interest (interview, commercial, etc.). For the learned baselines, we report the average F1 score and standard deviation over five random weight initializations.

Setup. For the image classification baseline (ResNet-50 Image Classification column in Table 2), we use a transfer learning approach [3] to fine-tune a ResNet-50 image classifier [50] (pre-trained on ImageNet) for each task. We also report the performance of this model after temporal “smoothing”: taking the mode

Task	Method				Rekall
	ResNet-50 Image Classification	ResNet-50 Image Classification + Smoothing	Conv3D Action Recognition	Rekall	
INTERVIEW	78.3 \pm 7.6	88.6 \pm 5.3	17.7 \pm 18.3	95.5	
COMMERCIAL	90.9 \pm 1.0	90.0 \pm 0.9	88.6 \pm 0.4	94.9	
CONVERSATION	65.2 \pm 3.5	66.1 \pm 3.5	79.4 \pm 2.3	71.8	
SHOT DETECT	—	—	83.2 \pm 1.0	84.1	
SHOT SCALE	67.3 \pm 1.0	68.1 \pm 1.2	70.1 \pm 0.8	96.2	
Faster R-CNN Object Detection					Rekall
PARKING	98.6 \pm 0.9	—	—	—	98.0

Table 2: We validate the accuracy of REKALL queries against learned baselines. For classification tasks (top five rows) we train image classification networks (with and without temporal smoothing) and a temporal action recognition network as baselines and report F1 scores. We cast PARKING as an object detection task and report average precision (AP) against a Faster R-CNN baseline. For all learned baselines, we report average scores and standard deviations over five random weight initializations.

of model predictions over a window of seven frames (ResNet-50 Image Classification + Smoothing). Since SHOT DETECT fundamentally requires information from multiple frames to detect a shot change, we did not run the ResNet-50 baselines for this task. For the action recognition baseline (Conv3D Action Recognition column), we fine-tune a ResNet34-based 3D CNN (pre-trained on the Kinetics action recognition dataset) for each task [20]. This network produces a single classification result given video segments of 16 frames. We chose these methods to represent a breadth of “reasonable-effort” learning approaches to these classification problems, with a range of temporal information available to the networks (the image classification baseline can only see a single image, whereas the smoothing aggregates signal from a small window, and the action recognition baseline is directly fed a larger window of frames).

Details of the experimental setup for each classification task are given below.

INTERVIEW: We limit the task to detecting interviews with Bernie Sanders (and any host). We annotated 54 hours of TV news broadcasts, split into a development set of 25 hours, and a test set of 29 hours. Interviews with Sanders are rare; the development set contains 40 minutes of Sanders interviews, and the test set contains 52 minutes.

COMMERCIAL: We annotated commercials in 46 hours of TV news broadcasts, and partitioned this video into a development set of 26 hours (7.5 hours of commercials), and a test set of 20 hours (6.3 hours of commercials).

CONVERSATION: We annotated conversations in 45 minutes of video selected from four films as a development set (29 minutes of conversations), and annotated all conversations in a fifth film (87 minutes of footage, 53 minutes of conversations) as a test set.

SHOT DETECT: We annotated shot boundaries in 45 minutes of video clips randomly selected from 23 movies, which we split in half into a development set with 348 shot transitions and a test set with 303 shot transitions.

SHOT SCALE: We annotated 898 shots generated by the SHOT DETECT query with scale annotations (293 long shots, 294 medium shots, and 311 close up shots). We split these in half into a development set and a test set.

Results. REKALL queries yielded a higher F1 score than the best learned model in four of the five classification tasks (6.1 F1 points greater on average across all five tasks). The largest difference in accuracy was for SHOT SCALE, where the REKALL query was 26.1 F1 points higher than the best learned approach.

The performance of different learned approaches varied widely by task; smoothing drastically improved the accuracy of the image classification model for INTERVIEW, but much less so for CON-

VERSATION and SHOT SCALE, and *decreased* the accuracy for COMMERCIAL. The action recognition baseline was the most accurate learning approach for both CONVERSATION and SHOT SCALE, but was less accurate than the image classification approaches for COMMERCIAL and 77.8 F1 points lower than the REKALL query for INTERVIEW.

The learned baselines in Table 2 were chosen as reasonable-effort solutions that follow common practice. It is likely that a machine learning expert, given sufficient time, could achieve better results. However, two of the tasks, COMMERCIAL and SHOT DETECT, are well-studied and have existing industrial or academic solutions. We compared our REKALL commercial detector against that of MythTV [2], an open-source DVR system. The MythTV detector achieved an F1 score of 81.5 on our test set, 14.0 F1 points worse than the REKALL query.

For SHOT DETECT, we compared against an open source implementation of the DeepSBD shot detector [21], trained on the large ClipShots dataset [57]. The DeepSBD shot detection model achieved an F1 score of 91.4, more accurate than our REKALL query.

However, by using the REKALL query’s output on our entire 589 movie dataset as a source of weak supervision [45–47], we are able to train a model that achieved an F1 score of 91.3, matching the performance of the DeepSBD model. *By using an imperfect REKALL query as a source of weak supervision on a large, unlabeled video database, we were able to train a model that matches the performance of a state-of-the-art method using 636× less ground truth data.* For more details on this approach, see the Appendix.

6.1.2 Object Detection Tasks

In PARKING we are interested both in detecting *when* there is an open parking spot, and *where* the open parking spot is in the frame. Therefore, we cast it as an object detection problem; given an image of a parking lot, detect all the empty parking spots. We gathered two hours of parking lot footage, annotated all the empty parking spots, and split it into a development and test set. We fine-tuned the Faster R-CNN model with ResNet-50 backbone [38, 48] (pre-trained on MS-COCO [37]) on the development set. The bottom row of Table 2 reports average precision (AP) for PARKING. Both the learned baseline and the REKALL query are near-perfect, achieving over 98.0 AP on this task.

6.1.3 Query Performance

Evaluation of deep neural network models often dominates the cost of video analysis, so many recent video analysis systems focus on accelerating (or avoiding) model execution [27, 28, 42]. In our application tasks, since the labels were pre-computed, REKALL

User ID	FP Experience	AP Scores	
		Original	Modified
1	5	78.2	98.7
2	4	75.0	98.7
3	3	74.2	98.0
4	3	66.5	95.5
5	3	65.9	94.2
6	2	66.5	95.5
7	1	26.5	95.5
8	1	0.0	0.0

Table 3: Results of a user study with novice users on the PARKING task. Participants were trained to use REKALL for one hour and then were given one hour to develop REKALL programs for the PARKING task. The Original column reports the average precision of their resulting programs. These scores were confounded by class confusion in the off-the-shelf object detector; scores after modification to account for this confusion (one LOC change for users 1-6, 3 LOC for user 7) are shown in the Modified column. Self-reported scores for familiarity with functional programming are shown in the FP Experience column.

queries were able to run over the development sets quickly enough to enable iterative query development (even though the REKALL implementation contains minimal optimization). Final queries for all tasks ran in less than thirty seconds on the development sets. The REKALL implementation stands to gain from further optimization, but even naive implementations of the composition functions were sufficient to provide interactive feedback to enable productive query development. Since initial model evaluation is the primary computational cost, future versions of REKALL might employ the various optimizations explored in video analysis systems from the literature to reduce the initial cost of pre-computing labels (or use query structure to guide cascades of model evaluation).

6.2 Usability

REKALL programs have been authored by students in two university research labs and at one company. Our experiences suggest that many programmers can successfully translate high-level video analysis tasks into queries.

For example, four of the representative tasks reported in Section 5 (COMMERCIAL, CONVERSATION, SHOT SCALE, FILM IDiom) were solved by domain experts who had no previous experience using REKALL. (*Note: These users are now associated with the project and are co-authors of this paper.*) These programming efforts ranged from an afternoon to two days (time to learn how to use REKALL), and included overcoming common query programming challenges such as dealing with noise in source annotations or misaligned transcript data.

These anecdotal experiences were promising, but the domain experts often developed REKALL queries in close collaboration with REKALL developers. To conduct a more quantitative assessment of the usability of REKALL, we ran a user study to evaluate how well novice users of REKALL were able to author queries to detect new events given one hour of query development time.

We recruited eight students with backgrounds ranging from medical imaging to machine learning and computer architecture. Participants received a one-hour REKALL training session, and then were given one hour to write a query for the PARKING task, along with a high-level English-language description of an approach similar to the one described in Section 5. Six of the eight students were able to successfully create a label set representing parking spaces, and use the `minus` operation to subtract car detections. Although

their queries were algorithmically similar to our results, none of the students was able to achieve comparable accuracy to our solution, since they did not account for cross-class confusion (cars being mis-classified as trucks). After modifying participant queries to account for these failures, user queries achieved average precision scores above 94.

The average precision scores of participant queries are shown in Table 3; the results from the original queries are shown in the Original column.

Users 1-6 were able to successfully subtract car detections from a label set representing parking spaces. Of those six, three (users 1-3) had enough time remaining to further iterate by using the `coalesce` operation to smooth over noise in the output of the object detector. This increased the AP scores of their algorithms by an average of 11.5 points. User 7 developed a query computing a `minus` operation between proposed parking spots and car detections, but did not correctly construct the parking spots event set; this resulted in an AP score of 26.5. User 8 failed to use REKALL to construct the initial parking spot label set, which was later determined to be due to a lack of familiarity with functional programming interfaces.

After the study, we manually modified each student’s solution to understand how far they were from optimal. The results from the modified queries are shown in the Modified column. Our modifications relaxed the `minus` operation to subtract out more objects, and not just the detected cars. For users 1-7, this was a single line change (equivalent of line 8 in Figure 7). For user 7, we additionally fixed the incorrect parking spot construction (equivalent of line 5 in Figure 7). After our changes, the queries written by users 1-7 all achieved AP scores greater than 94. The modified versions of the queries written by users 1 and 2 were more accurate than our algorithm for PARKING. This suggests that the participants were close, and with a better understanding of computer vision failure modes and debugging tools they could likely have achieved near-perfect accuracy.

Success using REKALL was strongly correlated with participants’ self-rated familiarity with functional programming. Before the tutorial, we asked users to rate their familiarity with functional programming on a scale of 1 – 5, with 1 being “Not at all familiar” and 5 being “Most familiar.” The self-reported scores are shown in column FP Familiarity in Table 3.

7. RELATED WORK

Multimedia Database Systems. The idea of associating database records with video intervals and composing queries for video search goes back to multimedia database systems from the 90’s and early 2000’s. Systems such as OVID, MMVIS, AVIS, CVQL, and BiL-Video aimed to provide an interface to query for complex events in video [4, 15, 23, 31, 32, 36, 40]. The query languages for these systems supported spatiotemporal joins based on Allen interval operations [5] and included operations to express repetitions of patterns.

However, queries written in these systems lacked the starting point of a large set of useful video annotations that modern machine learning technologies can provide. The modern machine learning landscape now makes it possible for heuristic composition to quickly define new events. We view our efforts as adapting early ideas about spatiotemporal composition to a modern context, with modern primitive annotations, large video datasets, and new, real-world video analysis tasks.

Domain-Specific Video Retrieval Systems. Our work is related to work on domain-specific video retrieval systems, such as SceneSkim or RoughCut in the film domain [34, 35, 39, 41, 49, 59–61] or Chalkboarding in the sports domain [43, 44, 52, 53]. These sys-

tems take advantage of domain-specific structure to support efficient video retrieval.

These approaches are largely complementary to programmatic composition; many of them break down video streams into domain-specific events, and allow users to query over sequences of those events. REKALL could be used to express or model these domain-specific events, and the broader patterns could be broken down into a series of composition operations.

Complex Event Processing and Diverse Analytics Systems.

REKALL’s composition operations takes inspiration from complex event processing and temporal stream analysis systems, such as Apache Flink, SiddiQL, and Microsoft Trill [8, 16, 25]. Adapting operators from the complex event processing systems to the video domain required support for a few language features that are not universal among complex event processing systems. In particular, we found that a continuous interval representation instead of a point representation was necessary to model data from different modalities, sampled at different resolutions. Complex event processing systems such as Apache Flink and SiddiQL are based on point events only, so they lack many of the interval-based operations that we have found necessary for video event specification. Trill provides a rich set of operations for analytics on data streams, including interval-based operations and anti-semi joins, so its expressivity is similar to REKALL. The one operation that does not appear directly is an equivalent of `coalesce`. It would be interesting to consider how systems like Trill could be adapted for future video event specification tasks.

Few-Shot and Zero-Shot Learning. Our approach is also related to recent efforts in few-shot and zero-shot learning [17, 24, 33, 55, 56, 62]. In fact, one way to view programmatic composition is as a mechanism for few-shot or zero-shot event detection in video, where the query programmer uses a small number of examples to guide query development. The mechanics of the approach to writing a REKALL query are different from most approaches to few-shot or zero-shot learning, but some of the principles remain the same; programmatic composition relies on information from pre-trained networks in order to compose them into complex events, for instance.

REKALL queries could also be used in conjunction with few-shot learning approaches. In many of our scenarios, REKALL queries are used for video event retrieval when there are no examples, only an idea in a developer’s head. In these cases, initial REKALL queries with human-in-the-loop curation could be used to source an initial small amount of labeled data to bootstrap few-shot learning approaches.

Weak Supervision Systems. We have shown that it is possible in a number of situations to use REKALL to write queries that accurately detect new events of interest, but it may not always be possible to programmatically specify new events accurately enough for all downstream applications. Weak supervision systems such as Snorkel [45–47] and Coral [58] use statistical techniques to build accurate models when accurate heuristics may be difficult to author. One use of REKALL is as a mechanism for writing richer heuristics for these weak supervision systems when they are operating over video; in fact, we utilize weak supervision techniques in the **Shot** task to weakly supervise a model that is more accurate than a model trained on $636\times$ more ground-truth data than we had access to. More work may be required to adapt weak supervision techniques to the video domain [30]; in the future, we plan on exploring how to more deeply integrate REKALL queries into weak supervision systems.

Video Retrieval Through Natural Language Interfaces. Some recent works at the intersection of computer vision and natural lan-

guage processing have centered around the task of action localization through natural language interfaces [18, 22]. These approaches aim to solve a similar problem to the video event specification problem that we consider, but the setup is slightly different; they are limited to the set of natural language descriptions found in the training/test distributions. This is fine if the events of interest are “in distribution,” but becomes problematic for more complex events. For example, the query “Jake Tapper interviews Bernie Sanders” would fail unless the network were trained on a broad enough set of queries to understand who Jake Tapper and Bernie Sanders were, and to create an embedding for the concept of an interview. The programmatic approach to event specification allows analysts to encode complex concepts directly by using domain knowledge to compose simpler concepts together.

8. DISCUSSION

REKALL is intended to give analysts a new tool for quickly specifying video events of interest using heuristic composition. Of course, the notion of authoring code in a domain-specific query language is not new, but adopting this approach for video analysis contrasts with current trends in modern machine learning, which pursue advances in video event detection through end-to-end learning from raw data (e.g. pixels, audio, text) [20]. Constructing queries through procedural composition lets users go from an idea to a set of video event detection results rapidly, does not incur the costs of large-scale human annotation and model training, and allows a user to express heuristic domain knowledge (via programming), modularly build on existing labels, and more intuitively debug failure modes.

However, compositional video event specification has many known limits. REKALL queries still involve manual parameter tuning to correctly set overlap or distance thresholds for a dataset. Higher-level composition is difficult when lower-level labels do not exist or fail in a particular context. (Our film studies efforts failed to build a reliable kissing scene detector because off-the-shelf face and human pose detectors failed due to occlusions present during an embrace.) In future work we plan to pursue REKALL extensions that model richer description of human behaviors or fine-grained movements, but there will always be video events that are less amenable to compact compositional descriptions and better addressed by learned approaches.

Nevertheless, we believe productive systems for compositional video event specification stand to play an important role in the development of traditional machine learning pipelines by helping engineers write programs that surface a more diverse set of training examples for better generalization, enabling search for anomalous model outputs (feeding active learning loops), or as a source of weak supervision to bootstrap model training. We hope that our experiences encourage the community to explore techniques that allow video analysis efforts to more effectively utilize human domain expertise and more seamlessly provide solutions that move along a spectrum between traditional query programs and learned models.

9. REFERENCES

- [1] Amazon rekognition. <https://aws.amazon.com/rekognition/>, 2019.
- [2] Mythtv, open source dvr. <https://www.mythtv.org/>, 2019.
- [3] Pytorch: Transfer leraning tutorial. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, 2019.
- [4] S. Adali, K. S. Candan, S.-S. Chen, K. Erol, and V. Subrahmanian. The advanced video information system:

- data structures and query processing. *Multimedia Systems*, 4(4):172–186, Aug 1996.
- [5] J. F. Allen. Maintaining knowledge about temporal intervals. *Communication of the ACM*, pages 832–843, 1983.
- [6] C. Brachmann, H. I. Chunpir, S. Gennies, B. Haller, T. Hermes, O. Herzog, A. Jacobs, P. Kehl, A. P. Mochtarraam, D. Möhlmann, et al. Automatic generation of movie trailers using ontologies. *IMAGE-Journal of Interdisciplinary Image Science*, 5:117–139, 2007.
- [7] K. L. Brunick, J. E. Cutting, and D. J. E. Low-level features of film: What they are and why we would be lost without them. *Psychocinematics: Exploring cognition at the movies*, pages 133–148, 2013.
- [8] B. Chandramouli, J. Goldstein, M. Barnett, R. DeLine, D. Fisher, J. Platt, J. Terwilliger, J. Wernsing, and R. DeLine. Trill: A high-performance incremental query processor for diverse analytics. VLDB - Very Large Data Bases, August 2015.
- [9] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1(3):275–316, 1997.
- [10] J. E. Cutting. The framing of characters in popular movies. *Art & Perception*, 3(2):191–212, 2015.
- [11] J. E. Cutting. The evolution of pace in popular movies. *Cognitive research: principles and implications*, 1(1), 2016.
- [12] J. E. Cutting and K. L. Armstrong. Facial expression, size, and clutter: Inferences from movie structure to emotion judgments and back. *Attention, Perception, & Psychophysics*, 78(3):891–901, 2016.
- [13] J. E. Cutting, K. L. Brunick, D. J. E., C. Irincinschi, and A. Candan. Quicker, faster, darker: Changes in Hollywood film over 75 years. *i-Perception*, 2(6):569–76, 2011.
- [14] J. E. Cutting and A. Candan. Shot durations, shot classes, and the increased pace of popular movies. *Projections: The Journal for Movies and Mind*, 9(2), 2015.
- [15] M. E. Dönderler, O. Ulusoy, and U. Güdükbay. Rule-based spatiotemporal query processing for video databases. *The VLDB Journal*, 13(1):86–103, Jan. 2004.
- [16] E. Friedman and K. Tzoumas. *Introduction to Apache Flink: Stream Processing for Real Time and Beyond*. O'Reilly Media, Inc., 1st edition, 2016.
- [17] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–932, 2016.
- [18] J. Gao, C. Sun, Z. Yang, and R. Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5267–5275, 2017.
- [19] A. Geitgey. Snagging parking spaces with mask r-cnn and python: Using deep learning to solve minor annoyances, 2019.
- [20] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.
- [21] A. Hassanien, M. Elgarib, A. Selim, S.-H. Bae, M. Hefeeda, and W. Matusik. Large-scale, fast and accurate shot boundary detection through spatio-temporal convolutional neural networks. *arXiv preprint arXiv:1705.03281*, 2017.
- [22] L. A. Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell. Localizing moments in video with natural language. In *International Conference on Computer Vision (ICCV)*, 2017.
- [23] S. Hibino and E. A. Rundensteiner. A visual query language for identifying temporal trends in video data. In *Proceedings International Workshop on Multi-Media Database Management Systems*, pages 74–81, Aug 1995.
- [24] M. Jain, J. C. van Gemert, T. Mensink, and C. G. Snoek. Objects2action: Classifying and localizing actions without any video example. In *Proceedings of the IEEE international conference on computer vision*, pages 4588–4596, 2015.
- [25] M. Jayasinghe, A. Jayawardena, B. Rupasinghe, M. Dayarathna, S. Perera, S. Suhothayan, and I. Perera. Continuous analytics on graph data streams using wso2 complex event processor. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16*, pages 301–308, New York, NY, USA, 2016. ACM.
- [26] C. S. Jensen and R. Snodgrass. Temporal specialization and generalization. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):954–974, 1994.
- [27] D. Kang, P. Bailis, and M. Zaharia. Blazeit: Fast exploratory video queries using neural networks. *arXiv preprint arXiv:1805.01046*, 2018.
- [28] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.
- [29] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia. Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*, 2018.
- [30] S. Khattar, H. O'Day, P. Varma, J. Fries, J. Hicks, S. Delp, H. Bronte-Stewart, and C. Ré. Multi-frame weak supervision to label wearable sensor data. In *Proceedings of the Time Series Workshop at ICML 2019*, June 2019.
- [31] M. Köprülü, N. K. Cicekli, and A. Yazici. Spatio-temporal querying in video databases. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems, FQAS '02*, pages 251–262, London, UK, UK, 2002. Springer-Verlag.
- [32] T. C. T. Kuo and A. L. P. Chen. A content-based query language for video databases. In *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 209–214, June 1996.
- [33] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.
- [34] M. Leake, A. Davis, A. Truong, and M. Agrawala. Computational video editing for dialogue-driven scenes. *ACM Trans. Graph.*, 36(4):130:1–130:14, July 2017.
- [35] B. Lehane, N. E. O'Connor, H. Lee, and A. F. Smeaton. Indexing of fictional video content for event detection and summarisation. *J. Image Video Process.*, 2007(2):1–1, Aug. 2007.
- [36] J. Z. Li, M. T. Özsü, and D. Szafron. Modeling of moving objects in a video database. *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 336–343, 1997.

- [37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [38] F. Massa and R. Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. <https://github.com/facebookresearch/maskrcnn-benchmark>, 2018. Accessed: July 15, 2019.
- [39] N. Mohamad Ali, A. F. Smeaton, and H. Lee. Designing an interface for a digital movie browsing system in the film studies domain. *International Journal of Digital Content Technology and Its Applications*, 5(9):361–370, 2011.
- [40] E. Oomoto and K. Tanaka. Ovid: design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643, Aug 1993.
- [41] A. Pavel, D. B. Goldman, B. Hartmann, and M. Agrawala. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST ’15, pages 181–190, New York, NY, USA, 2015. ACM.
- [42] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian. Scanner: Efficient video analysis at scale. *ACM Trans. Graph.*, 37(4):138:1–138:13, July 2018.
- [43] P. Power, J. Hobbs, H. Ruiz, X. Wei, and P. Lucey. Mythbusting set-pieces in soccer. *MIT SSAC*, 2018.
- [44] P. Power, H. Ruiz, X. Wei, and P. Lucey. Not all passes are created equal: Objectively measuring the risk and reward of passes in soccer from tracking data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1605–1613. ACM, 2017.
- [45] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the 44th International Conference on Very Large Data Bases (VLDB)*, Rio de Janeiro, Brazil, 2018.
- [46] A. J. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, 2019.
- [47] A. J. Ratner, C. M. D. Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016.
- [48] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [49] R. Ronfard and T. T. Thuong. A framework for aligning and indexing movies with their script. In *Multimedia and Expo, 2003. ICME’03. Proceedings. 2003 International Conference on*, volume 1, pages I–21. IEEE, 2003.
- [50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [51] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [52] L. Sha, P. Lucey, Y. Yue, P. Carr, C. Rohlf, and I. Matthews. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 336–347. ACM, 2016.
- [53] L. Sha, P. Lucey, S. Zheng, T. Kim, Y. Yue, and S. Sridharan. Fine-grained retrieval of sports plays using tree-based alignment of trajectories. *arXiv preprint arXiv:1710.02255*, 2017.
- [54] J. R. Smith, D. Joshi, B. Huet, W. Hsu, and J. Cota. Harnessing ai for augmenting creativity: Application to movie trailer creation. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1799–1808. ACM, 2017.
- [55] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.
- [56] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.
- [57] S. Tang, L. Feng, Z. Kuang, Y. Chen, and W. Zhang. Fast video shot transition localization with deep structured models. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2018.
- [58] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré. Inferring generative model structure with static analysis. In *Advances in neural information processing systems*, pages 240–250, 2017.
- [59] H.-Y. Wu and M. Christie. Analysing cinematography with embedded constrained patterns. In *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing*, WICED ’16, pages 31–38, Goslar Germany, Germany, 2016. Eurographics Association.
- [60] H.-Y. Wu, Q. Galvane, C. Lino, and M. Christie. Analyzing Elements of Style in Annotated Film Clips. In *WICED 2017 - Eurographics Workshop on Intelligent Cinematography and Editing*, pages 29–35, Lyon, France, Apr. 2017. The Eurographics Association.
- [61] H.-Y. Wu, F. Palù, R. Ranon, and M. Christie. Thinking like a director: Film editing patterns for virtual cinematographic storytelling. *ACM Trans. Multimedia Comput. Commun. Appl.*, 23, 2018.
- [62] H. Yang, X. He, and F. Porikli. One-shot action localization by learning sequence matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1450–1459, 2018.
- [63] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

10. APPENDIX

10.1 Supplemental Videos

We provide eight supplementary videos on our paper website (<http://www.danfu.org/projects/rekall-vldb2020/>):

- Film trailer for *Star Wars: Episode III - Revenge of the Sith*. We used REKALL queries to mine for clips matching eleven different film idioms, based on a viral video “How To Make a Blockbuster Movie Trailer.” We manually selected a few and edited them together into a trailer according to a template.
- Video montage of TV News interviews. We used our interview query to find interview clips with the 2016 presidential candidates. We randomly sampled 64 clips from the results and edited them into a video montage.
- Supercut of reaction shots from *Apollo 13*. We wrote a query to find reaction shots – shots where the character onscreen is shown reacting to another (offscreen) character’s dialogue. We ran the query on *Apollo 13* and put together a supercut of all the results.
- Supercut of action sequences. We wrote a query to find action sequences, and put together a supercut of a sample of the results.
- Supercut of “May The Force Be With You.” We wrote a query to find all the times when the phrase “May The Force Be With You” is said in the *Star Wars* films, and put together a supercut of all the results.
- Supercut of Hermione between Ron and Harry. We wrote a query to find the spatial pattern of Hermione Granger between Ron and Harry in the *Harry Potter* series, and put together a supercut of all the results.
- Film idiom: Intensification. We wrote a query to find instances of the “intensification” film idiom – where the shot scale gets monotonically larger on one or both characters throughout the course of a conversation. We selected one instance from the result set to show (from the movie *Spellbound*).
- Film idiom: Star Wide. We wrote a query to find instances of the “start wide” film editing idiom – where the first shot in a conversation shows all the characters in the conversation. We selected one instance from the result set to show (from the movie *Interstellar*).

10.2 Additional Task Details

In this section, we provide additional code listings and details for the COMMERCIAL, SHOT DETECT, CONVERSATION, and FILM IDIOM tasks discussed in Section 5.

10.2.1 Commercial Detection

A code listing for the COMMERCIAL query is shown in Figure 9. In our dataset, commercials are delineated by sequences of black frames. They often have mixed-case or missing transcripts, while the transcripts of non-commercial segments typically contain upper case text and distinct speaker markers (>>). The commercial detection query takes advantage of these dataset-specific heuristics to partition the video by sequences of black frames (lines 9-14), filter out segments that intersect with speaker markers (lines 16-22), and add in segments where the transcripts are mixed-case or missing (lines 28-40). The `filter_against` join, used in line 16, performs

```

1 # Commercial Query
2 transcript_words = rekall.ingest(transcript, 1D)
3 histograms = rekall.ingest(database.table("hists"), 1D)
4 entire_video = rekall.ingest(database.table("video"), 3D)
5
6 transcripts_with_arrows = transcript_words
7     .filter(lambda word: '>>' in word)
8
9 black_frame_segs = histograms
10    .filter(lambda i: i.histogram.avg() < 0.01)
11    .coalesce(predicate = time_gap < 0.1s, merge = time_span)
12    .filter(lambda i: i["t2"] - i["t1"] > 0.5s)
13
14 candidate_segs = entire_video.minus(black_frame_segs)
15
16 non_commercial_segs = candidate_segs
17    .filter_against(
18        transcripts_with_arrows,
19        predicate = time_overlaps)
20
21 commercial_segs = entire_video
22    .minus(non_commercial_segs.union(black_frame_segs))
23
24 commercials = commercial_segs
25    .coalesce(predicate = time_overlaps, merge = time_span)
26    .filter(lambda i: i["t2"] - i["t1"] > 10s)
27
28 lower_case_word_segs = transcript_words
29     .filter(lambda word: word.is_lowercase())
30     .coalesce(predicate = time_gap < 5s, merge = time_span)
31
32 missing_transcript = entire_video
33     .minus(transcript_words)
34     .filter(lambda i: 30 < i["t2"] - i["t1"] < 270)
35
36 commercials = commercials
37     .union(lower_case_word_segs)
38     .union(missing_transcript)
39     .coalesce(predicate = time_gap < 45s, merge = time_span)
40     .filter(lambda comm: comm["t2"] - comm["t1"] < 300s)

```

Figure 9: A Rekall query to retrieve commercials in a video collection of TV News broadcasts.

a left outer join and keeps any label from the left set that passes the predicate with any label from the right set. This query also makes heavy use of the `coalesce` function to connect sequences of labels that are temporally disconnected.

10.2.2 Shot Transition Detection

A code listing for the SHOT DETECT query is shown in Figure 10. This query uses color histograms of each frame to find sudden changes in the frame content. It computes the differences between histograms of neighboring frames and finds sudden spikes in the differences by computing the average and standard deviation of the change over a large window of frames. The query further reduces false positives by using the positions of face detections at the beginning and end of each shot transition; in short, if the query finds the same number of faces in the same positions at the beginning and end of a shot transition, it removes the transition as a false positive.

Lines 4-12 compute differences between the histograms in neighboring pairs of frames. Lines 14-28 construct a sliding window over the entire film, and compute the average and standard deviation of between-frame changes for each window. Lines 30-35 isolate frame pairs where the change in frame histogram is at least 2.5 standard deviations greater than the average change in the entire window, and remove transitions that are within 10 frames of

```

1 # Shot Query
2 histograms = rekall.ingest(database.table("hists"), 1D)
3
4 hist_pairs = histograms
5   .join(
6     histograms,
7     predicate = after(1 frame),
8     merge = time_span)
9
10 hist_diffs = hist_pairs
11   .map(lambda pair: pair.first
12       with payload = diff(pair.first, pair.second))
13
14 windows = hist_diffs
15   .join(
16     hist_diffs,
17     predicate = before or after 500 frames,
18     merge = lambda a, b: a with payload = [b])
19   .coalesce(
20     predicate = time_equal,
21     merge = lambda a, b:
22       a with payload = a.payload + b.payload)
23   .map(lambda window:
24     window with payload = {
25       diff: window.interval.diff,
26       avg: mean(window.payload),
27       stddev: stddev(window.payload)
28     })
29
30 transitions = windows
31   .filter(lambda i: i.diff > 2.5 * i.stddev + i.avg)
32   .coalesce(
33     predicate = time_gap < 10 frames,
34     merge = time_span)
35   .map(lambda t: t with t["t2"] = t["t1"] + 1 frame)
36
37 faces = rekall.ingest(database.table("faces"), 3D)
38
39 faces_by_frame = faces.groupby(lambda face: face["t1"])
40
41 transitions_starting_faces = transitions
42   .join(
43     faces,
44     predicate = start_time_equal,
45     merge = lambda transition, faces:
46       transitions with payload = faces)
47
48 transitions_ending_faces = transitions
49   .join(
50     faces,
51     predicate = end_time_equal,
52     merge = lambda transition, faces:
53       transitions with payload = faces)
54
55 bad_transitions = transitions_starting_faces
56   .join(
57     transitions_ending_faces,
58     predicate = time_equal and faces_match,
59     merge = lambda t1, t2: t1)
60
61 transitions = transitions.minus(bad_transitions)

```

Figure 10: A Rekall query to detect shot transitions in film.

any other transitions. Lines 41-53 associate faces with the beginning and end of each transition, and lines 55-59 use the custom `faces_match` predicate to detect transitions where the faces at the beginning and end of each transition are in the same place (just comparing the count and position of the nested face detections). Finally, line 63 removes the bad transitions from the pool.

For sake of exposition, we have presented this query as detecting

shot transitions in a film (i.e., the frame(s) at which a shot changes). This is how we evaluate the accuracy of a shot detector (how many of the detected transitions represent true transitions), but we store shot information in terms of continuous shots (i.e., we store a single label for each contiguous shot, instead of a label for each shot transition). The conversion is simple; a `minus` operation can partition a film into contiguous shots to be stored for future use (such as for the SHOT SCALE or CONVERSATION queries).

10.2.3 Conversation Detection

```

1 # Conversation Query
2 faces = rekall.ingest(database.table("faces"), 3D)
3 shots = rekall.ingest(database.table("shots"), 1D)
4
5 faces_by_frame = faces.groupby(lambda face: face["t1"])
6 shots_with_face_clusters = shots
7   .join(
8     faces_by_frame,
9     predicate = time_overlaps,
10    merge = span)
11   .groupby(lambda shot: (shot["t1"], shot["t2"]))
12   .map(lambda shot:
13     shot with shot.payload = cluster_faces(shot.faces))
14
15 shot_pairs = shots_with_face_clusters
16   .join(
17     shots_with_face_clusters,
18     predicate = lambda s1, s2: s1["t2"] == s1["t1"],
19     merge = span)
20
21 shot_pair_sequences = shot_pairs
22   .coalesce(
23     predicate = time_overlaps and
24       lambda shot_pair1, shot_pair2:
25         True if exists tup1 in shot_pair1.id_pairs,
26             tup2 in shot_pair2.id_pairs |
27             (dist(tup1[0], tup2[0]) < THRESHOLD and
28              dist(tup1[0], tup2[0]) < THRESHOLD) or
29              (dist(tup1[0], tup2[1]) < THRESHOLD and
30                dist(tup1[1], tup2[0]) < THRESHOLD),
31         merge = span
32   )
33
34 adjacent_sequences = shot_pair_sequences
35   .join(
36     shot_pair_sequences,
37     predicate = after one frame and
38       lambda shot_pairs1, shot_pairs2:
39         true if exists tup1 in shot_pairs1.id_pairs,
40             tup2 in shot_pairs2.id_pairs |
41             (dist(tup1[0], tup2[0]) < THRESHOLD and
42               dist(tup1[0], tup2[0]) < THRESHOLD) or
43               (dist(tup1[0], tup2[1]) < THRESHOLD and
44                 dist(tup1[1], tup2[0]) < THRESHOLD),
45         merge = span)
46
47 conversations = shot_pair_sequences
48   .union(adjacent_sequences)
49   .filter(num_shots >= 3)
50   .coalesce(predicate = time_overlaps, merge = span)

```

Figure 11: A Rekall query to detect conversations in film.

A code listing for the CONVERSATION query is shown in Figure 11. This query makes heavy use of a simple heuristic for detecting conversations – shots of the same people appearing multiple times in a contiguous film segment. In our dataset, caption data was difficult to obtain (manual searching and scraping repository websites), and was often unreliable (missing or badly mis-aligned with

the video files). As a result, we decided to write our query entirely from visual content. Since films are highly stylized, we couldn't rely on off-the-shelf identity recognition (which may have failed on minor characters). Instead, we used distance in face embedding space as a simple proxy for whether two detected faces belonged to the same character or not.

The query starts by loading up face detections and embeddings from off-the-shelf face detection and embedding networks [51,63], and contiguous shots output by the SHOT DETECT query (lines 2-3). It then clusters the faces in a single query by the face embeddings, using the custom `custom_faces` map function, which just uses K-means clustering with K equal to the maximum number of faces in a single frame in the shot (lines 5-13). The query next constructs pairs of neighboring shots (lines 15-19), and uses the `coalesce` function to construct sequences of shot pairs where the same people appear throughout the sequence, expressed by a custom predicate function (lines 21-32). Finally, the query joins together neighboring sequences, eliminates any sequences that are fewer than three shots long – i.e., the ones that are still shot pairs after the sequence construction (lines 47-50).

Conversation detection algorithm in Figure 11. English-language description of the steps here.

10.2.4 Film Idiom Mining

In this section, we present a full list of film idioms we mined for (including for the film trailer and supplementary videos), and give a brief description of the REKALL queries used to mine for these idioms. Many of these queries operate solely on face or object detections and transcript data, although a few also utilize identity information for main characters or the main villain. Before starting the mining process for the movie, we cluster the faces in a movie based on the face embeddings to quickly label all the main characters and the main villains in the movie – a process which took less than ten minutes per movie on average.

The film trailer template contained a total of six different film idioms that we mined for:

- Establishing shot – query searched for shots with no people or objects
- Ponderous statement or question – query searched the transcript for phrases like “what if?”
- Action sequences – query searched for sequences of multiple short shots in a row
- Dark tidings – query searched for sequences where the main villain was shown on screen along with some text in the transcript
- Hero shots – query searched for shots where a main character was displayed alone in a medium shot or close up, and the brightness of the frame was high (manually-set threshold)
- Shots of main characters looking hopeful – query was the same as the hero shots query, except allowing multiple characters in the frame

The trailer template contained a few other items as well, but we found that we could re-use existing queries for these. For example, the trailer template contains “statement of causality or finality from the main villain” as well as “dark tidings from the main villain,” but we were able to use the same dark tidings query for both.

10.3 Weak Supervision

In Section 6.1, we briefly discussed using REKALL queries as a source of weak supervision to train a model on a large collection of unlabeled data for the SHOT DETECT task. In this section, we discuss this approach in more detail.

The REKALL algorithm for the SHOT DETECT task was developed on a relatively small selection of data (half of 45 minutes of ground truth data that we collected), since collecting ground truth shot transition labels is expensive (labelers need to inspect every frame in a sequence to find the frame where a shot transition takes place). However, we had access to a much larger collection of unlabeled data – 589 feature-length films. We wanted to use the signal in the unlabeled data to improve over the performance of our REKALL query.

We turned to open-source *weak supervision* techniques in the academic literature such as Snorkel [45–47]. These techniques provide a mechanism to label a large collection of unlabeled data by using statistical methods to estimate the accuracies of different *labeling functions*. A labeling function is a function – any piece of code – that can label a data point. Weak supervision systems like Snorkel use the observed agreements and disagreements between labeling functions *on unlabelled data* to estimate the underlying accuracies of the labeling functions. The accuracies and labeling function outputs can then be used to output probabilistic labels over the unlabeled data, which can be used to train an end model using standard training techniques. To summarize, these weak supervision systems use multiple noisy user-defined labeling functions to generate probabilistic training labels for a collection of unlabeled data.

In order to apply these techniques to the SHOT DETECT task, we needed to turn our single query into multiple labeling function. The SHOT DETECT query can be viewed as a labeling function, but there is no way for weak supervision systems to observe agreements and disagreements between labeling functions if there is only a single labeling function.

To overcome this challenge, we broke the SHOT DETECT query into five different labeling functions – three that looked at changes in histograms between frames, and two that looked at the number and position of faces between frames. The three labeling functions that looked at changes in histograms between frames were all equivalent to lines 2-35 in Figure 10, but used different histograms as input – one used RGB histograms, one used HSV histograms, and one used optical flow histograms. The two face detection labeling functions, meanwhile, distilled the domain knowledge from lines 41-59 in Figure 10 and voted that there was no shot transition if they detected the same number of faces, or faces in the same position, respectively, across a series of frames. These labeling functions were all written in REKALL and generated noisy labels for the entire film dataset.

Once we generated these noisy labels, we used the weak supervision techniques in the Snorkel system [45] to learn the accuracies of the five labeling functions from their agreements and disagreements and generate probabilistic labels over the full unlabeled dataset. We then used these labels to train a shot detector, which achieved an F1 score of 91.3 on the held-out test set – matching the performance of another shot detector trained on a collection of gold ground-truth data 636× larger than our ground-truth dataset.