Bootstrapping Compositional Video Query Synthesis with Natural Language and Previous Queries from Users

Manasi Ganti
University of Washington
Seattle, USA
mganti@cs.washington.edu

Enhao Zhang
University of Washington
Seattle, USA
enhaoz@cs.washington.edu

Magdalena Balazinska
University of Washington
Seattle, USA
magda@cs.washington.edu

Abstract

With the emerging ubiquity of video data across diverse applications, the accessibility of video analytics is essential. To address this goal, some state-of-the-art systems synthesize declarative queries over video databases using example video fragments provided by the user. However, finding examples of what a user is looking for can still be tedious. This work presents POLY-VOCAL, a new system that eases this burden. POLY-VOCAL uses multiple forms of user input to bootstrap the synthesis of a new query, including textual descriptions of the user's search and previously synthesized queries. Our empirical evaluation demonstrates that POLY-VOCAL significantly improves accuracy and accelerates query convergence compared with query synthesis from only user-labeled examples, while lowering the effort required from users.

ACM Reference Format:
Manasi Ganti, Enhao Zhang, and Magdalena Balazinska. 2025. Bootstrapping Compositional Video Query Synthesis with Natural Language and Previous Queries from Users. In *Workshop on Human-In-the-Loop Data Analytics (HILDA' 25), June 22–27, 2025, Berlin, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3736733.3736738

1 Introduction

As video data becomes increasingly prevalent across domains such as traffic analytics, security, and civil engineering [10, 11, 28, 32, 33], there is a growing need to manage and query large video databases. Recent video database management systems (VDBMSs) allow users to retrieve specific events from large video datasets by specifying declarative queries [2, 3, 7, 23]. However, for users without database expertise, expressing compositional queries, where multiple objects interact in space and time (e.g., "find video fragments where a suspicious person is loitering outside the entrance of a building") using a declarative language can be difficult.

To address this challenge, some systems provide alternative ways for users to analyze and query videos [4, 17, 35, 40]. In particular, EQUI-VOCAL [40] provides a query-by-example interface, where users provide a few example video segments (for example, 10 positive and 10 negative examples) of their target event and the system automatically synthesizes a declarative query in a bottom-up fashion to find matching events. EQUI-VOCAL initializes its query synthesis process with an empty query and incrementally adds predicates, employing user labels to iteratively build a final query. While this approach eliminates the need for users to articulate declarative queries, it still has several limitations that we address in this paper.

First, users must manually locate positive examples of their target event to initiate query synthesis, which can be time-consuming and labor-intensive, especially as event specificity increases. Second, queries are synthesized from scratch, assuming no prior knowledge of the target event beyond a minimal set of initial examples. This can result in a lengthy learning process with many iterative steps. Lastly, intermediate queries synthesized in early iterations are often coarse-grained with low performance, making them difficult to disambiguate from other candidate queries. Labeling mistakes can further exacerbate errors in these early stages and propagate through later iterations, significantly derailing the final outcome. These challenges become more pronounced with longer and more complex target queries.

An alternative approach to using a video database management system would be to use a vision language model (VLM) [1, 20, 21, 27]. While VLMs are able to determine if a video fragment matches a natural language description, they struggle to answer compositional queries [14, 34]. Our goal in this paper is thus to leverage VDBMSs, but simplify the task of finding initial examples.

Besides providing labeled examples of the target event, there are alternative forms of user input that we can exploit to help the system synthesize queries more effectively. The growing capabilities of large language models (LLMs) for translating natural language (NL) descriptions into declarative queries present a potential solution to reducing user-labeling effort, as users can simply describe their target event in NL [18, 19, 29, 31]. However, NL is highly ambiguous, which can lead to inaccurate translations [6, 13, 25]. For example, a user analyzing security footage may provide the NL description "suspicious activity at entrance". This phrase is inherently ambiguous; it is unclear whether it refers to suspicious human or vehicle activity, and moreover, what constitutes as suspicious activity—loitering, breaking and entering, or something else. These ambiguities can make it challenging for LLMs to precisely map NL to the user's intended query. As we show in Section 4.3, directly executing the translated queries to find matching events results in poor query performance. Therefore, we aim to incorporate NL descriptions into the query synthesis process, but must address the challenge posed by NL ambiguity.

Another approach to guiding the initial steps of synthesis is to leverage domain knowledge about the iterative nature of exploratory video queries [8, 22, 36]. Users often start with simple queries to explore their data and then iteratively refine their queries to extract increasingly relevant results from the dataset. Consider

HILDA' 25, June 22–27, 2025, Berlin, Germany

ACM ISBN 979-8-4007-1959-2/2025/06
https://doi.org/10.1145/3736733.3736738

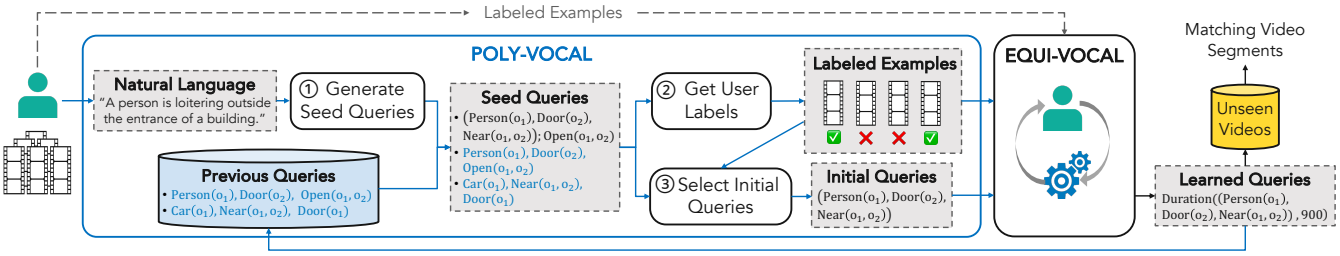


Figure 1: POLY-VOCAL pipeline. EQUI-VOCAL (gray dashed path) requires the user to manually provide labeled examples to synthesize a query from scratch. POLY-VOCAL (blue solid path) extends EQUI-VOCAL by incorporating natural language description and previously synthesized queries to ① generate seed queries, ② efficiently collect labeled examples from users, and ③ produce initial queries for subsequent synthesis by EQUI-VOCAL. All synthesized queries are stored for future use.

our running example: a user analyzing security footage for suspicious activity may start by looking for a person entering a building and later refine their search to find instances where a person is loitering outside the entrance of a building (but has not entered). In this scenario, the first query can inform the generation of the second. However, previous queries contain both relevant information (e.g., "person", "building") and extraneous predicates (e.g., "entering"), which must be identified and removed in order to leverage previous queries effectively.

In this paper, we introduce POLY-VOCAL, a system designed to bootstrap complex video query synthesis by incorporating various forms of user input to improve query performance and efficiency of the synthesis process while minimizing user effort. Extending the capabilities of EQUI-VOCAL, POLY-VOCAL learns user intent by leveraging NL descriptions and previously synthesized queries from users. POLY-VOCAL consists of three core components (Figure 1). First, POLY-VOCAL generates *seed queries* by utilizing LLMs to translate NL descriptions into compatible query notations for EQUI-VOCAL's synthesis process and by retrieving previously synthesized queries. Second, it utilizes seed queries to filter video subsets that are more likely to contain the target event for user labeling. Finally, instead of beginning the synthesis process with an empty query, POLY-VOCAL creates *initial queries*, based on seed queries derived from NL descriptions and previous queries, for subsequent query synthesis. Directly using LLM-translated or previously synthesized queries as starting points is problematic since they often contain irrelevant predicates that degrade performance. Instead, POLY-VOCAL enumerates subqueries from seed queries, ranking their effectiveness against labeled examples to select initial queries. By incorporating positive and negative examples, POLY-VOCAL clarifies the user intent expressed in NL.

POLY-VOCAL improves query performance and efficiency by starting query synthesis from initial queries, and reduces user effort by filtering a smaller set of candidate videos for user labeling. We evaluate the query performance, user effort, and synthesis efficiency of our approach on the CLEVRER dataset [37] with a curated set of 50 queries and show significant improvements on all fronts. Compared with vanilla EQUI-VOCAL, we raise the F1 score of the system by 22%, decrease the user effort by 60%, and accelerate query synthesis by 12%.

Table 1: Relational schema representation of data model.

`Objects`(vid, fid, oid, oname, x_1, y_1, x_2, y_2)
`Relationships`(vid, fid, rid, oid_1, rname, oid_2)
`Attributes`(vid, fid, oid, aname)

2 Background

POLY-VOCAL uses the data model and query synthesis pipeline of EQUI-VOCAL [40], which is based on spatio-temporal scene graphs [16]. In EQUI-VOCAL, a video is conceptually represented as three relations: `Objects`, `Relationships`, and `Attributes`. Queries over videos can thus be expressed as relational queries over the view consisting of these three relations (as shown in Table 1). However, queries in EQUI-VOCAL are not arbitrary SQL queries. Since these queries focus on how objects interact in space and time, EQUI-VOCAL restricts the set of relational queries that it supports to ensure efficient query synthesis. These queries are captured by a concise domain-specific language (DSL). For example, a query searching for instances where "a person is outside a building for at least 30 seconds, then the person enters the building" can be expressed as (assuming 30 frames per second): $\texttt{Duration}((\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Near}(o_1, o_2)), 900); \texttt{Open}(o_1, o_2)$ The variable o_i in a query is bound to an arbitrary object from the video, with distinct subscripts indicating objects with different identifiers. Within the same frame, predicates are separated by commas, forming a region graph. Region graphs are connected in temporal sequence with semicolons. The duration constraint $\texttt{Duration}(g, d)$ specifies the minimum number of frames d for which a region graph g persists.

To synthesize queries with EQUI-VOCAL, the user provides (i) a video dataset, (ii) user-defined functions (UDFs) that extract semantic information from videos to populate the relations in Table 1, and (iii) labeled examples—positive examples where the target event occurs and negative examples where it does not—as shown by the dashed path in Figure 1. EQUI-VOCAL begins with an empty query and progressively expands it by adding new predicates, leveraging user-provided examples to steer its search toward the target query.

3 POLY-VOCAL Approach

POLY-VOCAL reduces the difficulty of providing initial labeled examples by leveraging two alternative methods for capturing user

Algorithm 1: Query initialization and synthesis in POLY-VOCAL

Input : U - set of unlabeled video segments,
P - set of user-defined functions,
T - user's natural language description of target event,
Q_P - previously generated queries,
k_i - number of initial queries, hyperparameter
Output : Q_t - set of top synthesized queries

1 $Q_s \leftarrow Q_p$
2 **if** T **then**
3 $\quad Q_s \leftarrow Q_s \cup \text{LLMTranslate}(T, P)$
4 $L \leftarrow \text{GetUserLabels}(U, Q_s)$
5 $Q_0 \leftarrow \text{SelectInitialQueries}(L, Q_s, k_i)$
6 $Q_t \leftarrow \text{QuerySynthesis}(U, L, P, Q_0)$ `// Invoke EQUI-VOCAL`
7 **return** Q_t

intent: NL descriptions and previously synthesized queries. Algorithm 1 outlines this approach. The algorithm takes as input a set of unlabeled video segments (U), a set of user-defined functions for EQUI-VOCAL language specification (P), an NL description (T), and a set of previous queries (Q_p).

The algorithm consists of three main steps, as depicted by the blue solid path from the user in Figure 1. In Step ①, the user first specifies an NL description (T) of the target event that POLY-VOCAL, using an LLM, will translate into a query in the EQUI-VOCAL DSL. To supplement this, POLY-VOCAL will also fetch previously synthesized queries (Q_p). We refer to these queries derived from different sources of user intent as seed queries (Q_s) (Lines 1 to 3; see Section 3.1). POLY-VOCAL requires a non-empty set of seed queries to initialize synthesis. Lacking these, it falls back to the EQUI-VOCAL approach.

In Step ②, POLY-VOCAL executes a single seed query on the video database and shows the query results to the user. Unlike a naïve approach, where users manually search the entire dataset for examples of their target event, POLY-VOCAL shrinks the user's labeling effort by enabling users to efficiently locate initial positive and negative video fragments (L) within this smaller, pre-filtered set of query results (Line 4; see Section 3.2).

Since seed queries may not always align with the target event, directly employing them as initial inputs for synthesis can lead to low performance. In Step ③, POLY-VOCAL addresses this issue by leveraging the user labels to refine and prune the seed queries, producing a set of k_i initial queries (Q_0) of higher quality for EQUI-VOCAL's query synthesis process (Line 5; see Section 3.3).

POLY-VOCAL bootstraps the query synthesis process by efficiently obtaining labeled examples (L) and generating initial queries (Q_0). EQUI-VOCAL then takes these inputs, along with U and P, to synthesize a query for the target event (Line 6). The synthesized query is executed to find matching events in unseen videos. All user-accepted synthesized queries are stored for future use.

3.1 Generating Seed Queries

POLY-VOCAL begins by asking the user for an NL description of their target event. POLY-VOCAL utilizes an LLM to translate the user's description into a query written in the DSL (Algorithm 1, Line 3). Although the LLM does not directly access the video data, the LLM is provided with the DSL syntax definition and a list of valid predicates with semantic descriptions to guide this translation. This helps the LLM generate meaningful queries with valid predicates. The quality of the queries will likely vary depending on the LLM. A detailed example of this prompt can be found in [41]. LLM-translated queries can have syntax errors. To address this, POLY-VOCAL verifies the syntactic correctness of a generated query. If the generated query contains parsing errors, which can include misplaced parentheses or invalid predicates, POLY-VOCAL retries the generation process, incorporating feedback about the syntax error into the prompt for the next attempt until a syntactically correct query is produced. If the query is still incorrect after three tries, the user is asked to re-articulate their target event. In our experiments, we found that no seed query required more than two iterations of syntax correction. As improving the initial translation is not the focus of this paper, we assume that we start with a syntactically correct seed query resulting from this process.

The generated query is then combined with all past queries that the user has issued to create the set of seed queries.

3.2 Getting User Labels

While labeling examples of the target event is generally simpler than composing a declarative query, filtering through a large input dataset is often challenging, particularly when the target event is complex or rare. To address this, we utilize seed queries to extract a smaller, more promising subset of videos more likely to contain instances of the target event (Algorithm 1, Line 4). This subset is then presented to the user for labeling.

When multiple seed queries are present, only one is executed against the dataset to retrieve a relevant subset of videos for labeling. We employ the following heuristics: The LLM-generated query is used if available; otherwise, the most recent previous query is used. The LLM-generated query is prioritized because it is more likely to better align with the user's intended target event. When only previous queries are available, determining the most relevant one is difficult without additional user input, so we assume recency implies relevance. An alternative solution is to explicitly prompt the user to select from a list of previous queries. POLY-VOCAL assumes that seed queries, even when partially inaccurate, can filter the dataset to yield a pool with a higher proportion of positive samples than the overall corpus. For instance, if the user is searching for videos where a person loiters outside a building with the DSL representation "$\texttt{Duration}((\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Near}(o_1, o_2)), 900)$", a seed query such as "$(\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Near}(o_1, o_2)); \texttt{Open}(o_1, o_2)$" is not perfect. However, it could still capture video segments where a person is initially outside a building. The resulting filtered video set likely contains a higher proportion of positives than the original corpus due to the correct predicates "$\texttt{Person}(o_1)$", "$\texttt{Door}(o_2)$", and "$\texttt{Near}(o_1, o_2)$", enabling more efficient example collection.

Although a seed query is guaranteed to be syntactically correct, it may not always be semantically accurate. Semantic inaccuracies can prevent users from finding sufficient positive examples if relevant instances are filtered out by unrelated predicates within the seed query. If the number of positive examples remains insufficient after labeling all filtered videos, POLY-VOCAL prunes predicates from the seed query until enough positive examples are acquired. To determine which predicate to remove, POLY-VOCAL generates n query candidates, each omitting a different predicate from the

query, and selects the candidate that maximizes the filtered pool size. In our example, this would ideally lead to selecting the subquery "$(\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Near}(o_1, o_2))$", removing the most restrictive predicate "$\texttt{Open}(o_1, o_2)$".

While these assumptions may not hold universally, we empirically validate the effectiveness of our approach in Section 4. Future work could explore adaptive mechanisms to determine the optimal order of predicate removal.

3.3 Selecting Initial Queries

Ideally, seed queries closely match target events and can effectively serve as initial queries for query synthesis or directly retrieve matching videos. However, seed queries—either LLM-generated or previously synthesized—often contain extraneous predicates. For example, if the user intends to capture events where a person is loitering near a building entrance, an accurate DSL representation might be "$\texttt{Duration}((\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Near}(o_1, o_2)), 900)$". However, the user may provide the NL description "suspicious activity at entrance" and the LLM might incorrectly translate this into "$(\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Open}(o_1, o_2))$", retrieving footage of a person entering a building rather than loitering. Since the translated DSL query contains an erroneous predicate "$\texttt{Open}(o_1, o_2)$", using it as a starting point for query synthesis can introduce compounding errors, resulting in progressively less accurate queries.

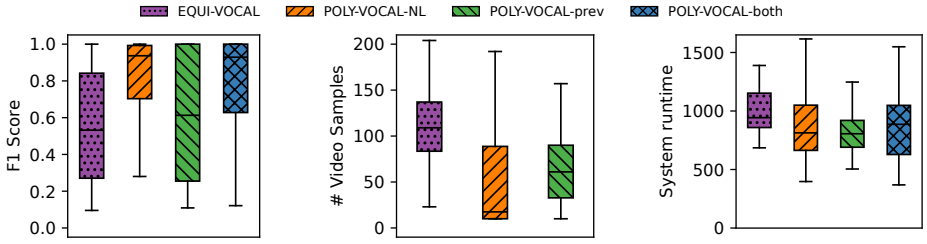
To synthesize target queries from initial queries effectively, the process must balance the retention of useful information with the removal of erroneous or extraneous predicates. One possible approach is to remove extraneous predicates during synthesis. However, since the EQUI-VOCAL synthesis process builds the query by modifying one predicate at a time, allowing predicate removal can lead to oscillations, where predicates are repeatedly added and removed across iterations. To prevent this, POLY-VOCAL incorporates predicate removal prior to query synthesis. Its goal is to craft an initial query that is a subquery of the target query—without extraneous predicates and with minimal missing predicates. More specifically, if the initial query is simpler or more generic than the actual target event, it can still guide query synthesis in the right direction, albeit at the cost of increased synthesis time. Conversely, if the initial query is entirely incorrect or overly complex, it can negatively impact query synthesis due to the presence of extraneous predicates. Therefore, correct queries are preferred over generic ones, while generic queries are preferred over incorrect ones.

To produce high-quality initial queries, POLY-VOCAL augments seed queries by also considering their subqueries, which contain the exhaustive combinations of the predicates within the query. For instance, the seed query "$(\texttt{Person}(o_1), \texttt{Door}(o_2), \texttt{Open}(o_1, o_2))$" will have enumerated subqueries including "$\texttt{Person}(o_1)$", "$\texttt{Door}(o_1)$", "$(\texttt{Person}(o_1), \texttt{Door}(o_2))$", etc. Following this step, our set of candidate seed queries is composed of (i) different previous queries and their subqueries as well as (ii) the NL translation and its subqueries. We utilize previously gathered user labels (Algorithm 1, Line 4) to evaluate the effectiveness of each candidate seed query (Line 5) and determine initial queries. We rank candidate seed queries based on F1 scores on those user labels and select the top-k_i queries, breaking ties randomly, where k_i is a hyperparameter.

After selecting initial queries, we start the EQUI-VOCAL synthesis process. Instead of expanding the single, empty query in the first iteration, EQUI-VOCAL now enumerates candidate queries by expanding all k_i initial queries, with subsequent steps—which retain and similarly expand the most promising candidates—remaining unchanged. A common failure in the original EQUI-VOCAL approach occurs in early iterations when ancestor queries of the target query are indistinguishable from other candidate queries with similarly low performance [40]. By starting from non-empty initial queries, POLY-VOCAL effectively prunes the search space and alleviates the issue by jumping directly to more specific ancestor queries.

4 Evaluation

Dataset: We evaluate our system on the CLEVRER dataset [37], comprising 10,000 five-second videos of moving 3D shapes. Following EQUI-VOCAL [40], we use 13 predicates covering spatial relationships, locations, colors, shapes, and materials. We curate a test set of 50 DSL queries with varying complexity, each executed to generate ground truth labels. An example query written for the dataset is "$(\texttt{Sphere}(o_1), \texttt{Sphere}(o_2), \texttt{LeftOf}(o_1, o_2)); \texttt{RightOf}(o_1, o_2)$".
NL descriptions: To evaluate the robustness of POLY-VOCAL to linguistic variability, we create three different descriptions for each DSL query: *Accurate description* closely aligns with the target event, e.g., "A sphere is to the left of another sphere, then it moves to its right"; *Generic description* is simpler yet generally oriented toward the target event, e.g., "A sphere changes position relative to another sphere"; *Incorrect description* contains incorrect or misleading details, e.g., "A sphere is to the left of another sphere, then it moves to the right quadrant". For the end-to-end experiment, one variation is randomly chosen as the NL description for each target query.
Previous queries: We simulate an exploratory query workload by generating a "previous query" for each target query, assuming that the user has previously executed this query prior to issuing the target query. Each previous query is created by randomly removing one or more predicates from the corresponding target query and introducing an additional predicate that is not present in the target query. For example, a previous query for the example above could be "$\texttt{Sphere}(o_1), \texttt{Red}(o_1), \texttt{LeftOf}(o_1, o_2)$".
Metrics: To measure query performance, we compute the F1 score between retrieved results and ground truth data. To measure user effort, we record the number of samples required to obtain sufficient positive examples. To measure efficiency, we record the system runtime of the entire pipeline to synthesize a query.
Baselines: We compare our system against the original EQUI-VOCAL system, which utilizes only user labels and does not use NL input or previous queries. For our approach, we consider three settings: POLY-VOCAL-NL uses only NL descriptions, POLY-VOCAL-prev uses only previous queries, and POLY-VOCAL-both uses both.
Experimental setup. Our experiments use the GPT-4o model (`gpt-4o-2024-08-06`) as the LLM. POLY-VOCAL requests user labels to collect at least 10 positive examples for each query. We use 500 videos from CLEVRER as training data for running POLY-VOCAL and EQUI-VOCAL and the rest as test data to evaluate the performance of synthesized queries. We set $k_i = 5$ for initial query selection. For query synthesis, we set the following EQUI-VOCAL hyperparameters: beam width $b_w = 5$, maximum number of predicates $n_{\text{pred}} = 10$.



(a) F1 Score (b) User Effort (c) Runtime

Figure 2: End-to-end performance of EQUI-VOCAL vs. POLY-VOCAL under three input settings.

4.1 End-to-End Performance

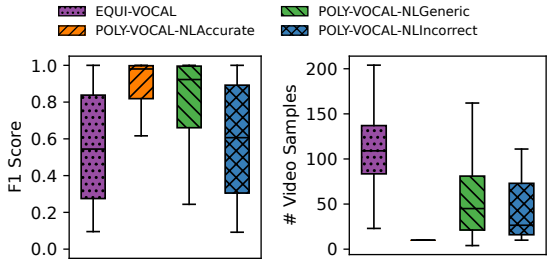
4.1.1 Query Performance. Figure 2a shows the F1 scores for the final synthesized queries on different systems. EQUI-VOCAL achieves a mean F1 score of 0.56. By generating initial queries for synthesis, POLY-VOCAL mitigates synthesis errors during early iterations and significantly improves query performance by up to 22% to 0.78 (with POLY-VOCAL-NL). In the absence of NL description, using prior synthesized queries offers modest gains, with POLY-VOCAL-prev improving the F1 score to 0.62. Interestingly, the F1 score of POLY-VOCAL-both drops slightly from 0.78 to 0.77 compared to POLY-VOCAL-NL. While POLY-VOCAL-both outperforms POLY-VOCAL-NL on simpler queries, i.e., queries with fewer than 5 predicates, as query complexity increases, POLY-VOCAL-NL performs better, leading to the decrease in average F1 score. This suggests that NL descriptions in our evaluation are typically more accurate than previous queries for complex queries, whereas prior queries are more helpful with simpler target events where their overlap with the target query is high.

4.1.2 User Effort. Figure 2b compares the user effort required to obtain 10 positive examples for initiating query synthesis across EQUI-VOCAL and POLY-VOCAL variants. We omit POLY-VOCAL-both from this comparison, since only the NL description is used to get user labels under this setting, making it functionally equivalent to POLY-VOCAL-NL. In EQUI-VOCAL, the user has to examine an average of 114 videos to find sufficient positive examples. POLY-VOCAL reduces this burden by 60%, requiring users to view only 45 video segments when using NL descriptions for filtering. When NL description is unavailable and a previously synthesized query is used as a seed query, the user effort is 46% lower than EQUI-VOCAL at 61 examples. This demonstrates that utilizing seed queries to filter the input dataset effectively addresses the bottleneck of manually searching for target event instances, thus reducing user effort.

4.1.3 Runtime. Figure 2c presents the system runtime for each variation of POLY-VOCAL. While EQUI-VOCAL has an average runtime of 1011 seconds, POLY-VOCAL-NL, POLY-VOCAL-prev, and POLY-VOCAL-both reduce the runtime average by 18, 15, and 12% to 836, 865, and 886 seconds respectively. POLY-VOCAL demonstrates improved runtime. By leveraging initial queries to accelerate the query synthesis process, it reduces the number of synthesis iterations required for each system to build the final query.

4.2 Natural Language Input

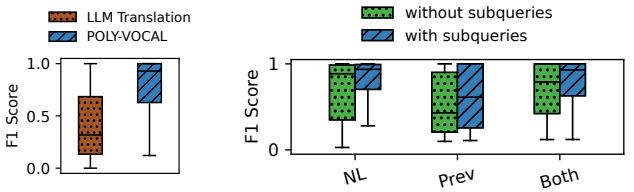
We evaluate the robustness of POLY-VOCAL on NL descriptions of



(a) F1 Score (b) User Effort

Figure 3: User effort and query performance of EQUI-VOCAL and POLY-VOCAL with three types of natural language input.

varying quality. For each target query, we provide accurate, generic, and incorrect descriptions, respectively, as input to POLY-VOCAL-NL. Figure 3a shows the F1 scores of POLY-VOCAL-NL with different qualities of NL descriptions and EQUI-VOCAL. We find that leveraging NL particularly helps improve the F1 score of the final query when the user's NL description is well-aligned, resulting in an average F1 score of 0.86 compared to EQUI-VOCAL's 0.56. When the user provides a more generic version of the query they seek, the NL description continues to serve as a good foundation to build on and improves performance to an F1 score of 0.80. Even with incorrect descriptions, POLY-VOCAL-NL slightly improves the F1 score to 0.60, as the initial query selection process ranks subqueries of seed queries and thus effectively removes incorrect predicates. Figure 3b shows the number of samples required to gather 10 positive examples. POLY-VOCAL significantly reduces user effort across all variations of NL descriptions. With accurate descriptions, all filtered videos are positive samples and thus POLY-VOCAL requests as few as 13 labels on average, an 89% reduction from the baseline of 114 examples. When the user's description is more generic, the user must look through an average of 60 videos—there are more potential positive examples for the user to sift through, but still fewer compared to no filtering. Notably, with incorrect descriptions, POLY-VOCAL reduces labeling effort to an average of 44 videos, outperforming generic descriptions. The reasons are twofold. First, generic queries can potentially introduce many false positives into the filtered set, thereby increasing the labeling burden. Second, incorrect descriptions typically lead to false negatives and significantly shrink the filtered pool, in which case the user fails to find enough positives even after labeling all filtered videos. Specifically,



(a) POLY-VOCAL vs. direct LLM translation. **(b) POLY-VOCAL without vs. with subquery enumeration.**

Figure 4: Comparison of initial query selection strategies.

11 out of 50 seed queries derived from incorrect descriptions are modified while getting user labels, and predicates are removed as described in Section 3.2. Nonetheless, these findings indicate that even imperfect NL input contains valuable information that POLY-VOCAL can effectively exploit to reduce user labeling effort.

4.3 Selecting Initial Queries

Given that LLMs can directly translate NL descriptions into DSL queries, one might question whether the translated query alone could serve as the final query without undergoing further query synthesis. To answer this question, we compare POLY-VOCAL with this direct LLM method. As illustrated in Figure 4a, POLY-VOCAL achieves an F1 score of 0.77—over 80% higher than the direct LLM method's 0.41—by incorporating the DSL query into the synthesis process. This indicates that relying solely on LLM-generated queries is insufficient, as NL descriptions provided by users may not accurately represent the target event. By integrating the translated DSL query with EQUI-VOCAL's query synthesis, POLY-VOCAL clarifies the user's intent and achieves significantly better results.

To validate the effectiveness of our query enumeration strategy in initial query selection, we evaluate the performance of three POLY-VOCAL variants, comparing results with and without query enumeration. As illustrated in Figure 4b, enumerating subqueries consistently improves F1 scores by effectively eliminating extraneous predicates from seed queries. The improvement is most pronounced for POLY-VOCAL-prev, where F1 scores increase by 20% (from 0.51 to 0.62). Previous queries benefit more from query enumeration because they typically exhibit higher degrees of misalignment with target queries.

5 Related Work

Compositional Video Query Processing. There are many systems for video analytics specifically designed for compositional queries [2, 3, 7, 23]. However, these systems require users to specify their queries or train a classifier for the purpose of identifying their target event, both of which are non-trivial for non-experts and can be time consuming. POLY-VOCAL utilizes more intuitive user input—the query-by-example approach inherited from EQUI-VOCAL, the user's own textual description of their query, and any queries previously synthesized by the user. These require minimal effort and expertise.

Text-to-Declarative-Query Translation. LLMs have been applied to Text-to-SQL and other Text-to-Declarative-Query translation tasks [19, 31] with impressive results on academic benchmarks [30, 38]. However, these benchmarks fail to capture the ambiguity of real NL input, which can degrade the quality of LLM-generated queries [12, 25]. Additionally, the need for extensive context—such as schema definitions and DSL constraints—can further challenge LLM performance [6, 18]. POLY-VOCAL mitigates these limitations by leveraging user-labeled examples to disambiguate the user intent expressed in NL.

Query Reuse. Reusing results in traditional DBMSs is a well-researched topic [5, 8, 9, 15, 24, 26], primarily focusing on optimizing query execution through exact syntactic matching [36]. However, such techniques are not well-suited for flexible reuse in exploratory or evolving query synthesis. POLY-VOCAL extends query reuse beyond syntactic equivalence, extracting valuable semantic information despite potential syntactic mismatches.

6 Limitations and Challenges

While POLY-VOCAL greatly reduces query synthesis time and user labeling effort, further improvements are needed to make it a more interactive and practical query-by-example framework. Currently, synthesizing one query with POLY-VOCAL-both requires 14.7 minutes, excluding user labeling time. While runtime can be reduced through hyperparameter tuning (e.g., smaller beam width, fewer examples), this creates performance trade-offs.

In our experiment, we request 10 positive examples to initiate query synthesis, requiring users to examine an average of 45 videos when using POLY-VOCAL with NL descriptions. While simpler queries may achieve good performance with as few as two initial positive examples [40], insufficient labeled examples can compromise the reliability of F1 scores used to evaluate candidate query quality. Moreover, labeling effort depends heavily on the quality of seed queries. For example, accurate NL descriptions can greatly reduce labeling requirements (Figure 3b), which highlights the need for better user guidance in formulating initial queries.

Future work could explore enhanced user interactivity throughout the POLY-VOCAL pipeline. To speed up synthesis, one promising direction is to enable users to inspect intermediate queries after each iteration [39] and terminate the process early once they are satisfied with the results. Furthermore, to improve the alignment of seed queries, we can present users with multiple previous queries and explicitly ask them to pick the most relevant ones.

7 Conclusion

In this paper, we introduced POLY-VOCAL, a novel approach to improving the query synthesis process by leveraging both NL descriptions and previously synthesized queries as seed queries. We ensure that even when user input is inaccurate, valuable structural and semantic information can still be extracted and refined to guide synthesis toward an accurate final query. Overall, POLY-VOCAL makes query generation more user-friendly, efficient, and robust.

Acknowledgments

This work was funded in part by NSF award 2211133.

References

[1] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. arXiv:2308.12966 [cs.CV]

[2] Favyen Bastani, Oscar R. Moll, and Samuel Madden. 2020. Vaas: Video Analytics At Scale. *PVLDB* 13, 12 (2020), 2877–2880.

[3] Daren Chao, Nick Koudas, and Ioannis Xarchakos. 2020. SVQ++: Querying for Object Interactions in Video Streams. In *SIGMOD*. 2769–2772.

[4] Yueting Chen, Nick Koudas, Xiaohui Yu, and Ziqiang Yu. 2022. Spatial and Temporal Constrained Ranked Retrieval over Videos. *PVLDB* 15, 11 (2022), 3226–3239.

[5] Kayhan Dursun, Carsten Binnig, Ugur Cetintemel, and TIm Kraska. 2017. Revisiting Reuse in Main Memory Database Systems. In *SIGMOD*. 1275–1289.

[6] Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex Van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, K. Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. 2024. NL2SQL is a solved problem... Not!. In *CIDR*.

[7] Daniel Y. Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong, Avanika Narayan, Maneesh Agrawala, Christopher Ré, and Kayvon Fatahalian. 2019. Rekall: Specifying Video Events using Compositions of Spatiotemporal Labels. arXiv:1910.02993 [cs.DB]

[8] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2017. Revisiting Reuse for Approximate Query Processing. *PVLDB* 10 (2017), 1142–1153.

[9] Jonathan Goldstein and Per-Åke Larson. 2001. Optimizing Queries Using Materialized Views: A practical, scalable solution. In *SIGMOD*. 331–342.

[10] Jie Gong and Carlos H. Caldas. 2010. Computer Vision-Based Video Interpretation Model for Automated Productivity Analysis of Construction Operations. *J. Comput. Civ. Eng.* 24, 3 (2010), 252–263.

[11] Patrick Hammer, Tony Lofthouse, Enzo Fenoglio, Hugo Latapie, and Pei Wang. 2020. A Reasoning Based Model for Anomaly Detection in the Smart City Domain. In *IntelliSys (AISC, Vol. 1251)*. 144–159.

[12] Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. arXiv:2106.05006 [cs.CL]

[13] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. arXiv:2406.08426 [cs.CL]

[14] Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. 2023. SugarCrepe: Fixing Hackable Benchmarks for Vision-Language Compositionality. In *NeurIPS*.

[15] Milena Ivanova, Martin L. Kersten, Niels J. Nes, and Romulo Goncalves. 2009. An architecture for recycling intermediates in a column-store. In *SIGMOD*. 309–320.

[16] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. 2020. Action Genome: Actions As Compositions of Spatio-Temporal Scene Graphs. In *CVPR*. 10233–10244.

[17] Ranjay Krishna, Vincent S. Chen, Paroma Varma, Michael S. Bernstein, Christopher Ré, and Li Fei-Fei. 2019. Scene Graph Prediction With Limited Labels. In *ICCV*. 2580–2590.

[18] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *NeurIPS*.

[19] Yilin Li and Deddy Jobson. 2024. LLMs as an Interactive Database Interface for Designing Large Queries. In *HILDA*. 1–7.

[20] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. In *NeurIPS*.

[21] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. 2024. DeepSeek-VL: Towards Real-World Vision-Language Understanding. arXiv:2403.05525 [cs.AI]

[22] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[23] Stephen Mell, Favyen Bastani, Steve Zdancewic, and Osbert Bastani. 2023. Synthesizing Trajectory Queries from Examples. In *CAV*, Vol. 13964. 459–484.

[24] Fabian Nagel, Peter A. Boncz, and Stratis Viglas. 2013. Recycling in pipelined query evaluation. In *ICDE*. 338–349.

[25] Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2024. Evaluating Ambiguous Questions in Semantic Parsing. In *ICDEW*. 338–342.

[26] Luis Leopoldo Perez and Christopher M. Jermaine. 2014. History-aware query optimization with materialized intermediate views. In *ICDE*. 520–531.

[27] Francisco Romero, Caleb Winston, Johann Hauswald, Matei Zaharia, and Christos Kozyrakis. 2023. Zelda: Video Analytics using Vision-Language Models. arXiv:2305.03785 [cs.DB]

[28] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. 2016. Planning for Autonomous Cars that Leverage Effects on Human Actions. In *Robotics: Science and Systems*.

[29] Dian Shao, Yu Xiong, Yue Zhao, Qingqiu Huang, Yu Qiao, and Dahua Lin. 2018. Find and Focus: Retrieve and Localize Video Events with Natural Language Queries. In *ECCV*, Vol. 11213. 202–218.

[30] Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2024. A Survey on Employing Large Language Models for Text-to-SQL Tasks. arXiv:2407.15186 [cs.CL]

[31] Ruoxi Sun, Sercan Arik, Rajarishi Sinha, Hootan Nakhost, Hanjun Dai, Pengcheng Yin, and Tomas Pfister. 2023. SQLPrompt: In-Context Text-to-SQL with Minimal Labeled Data. In *EMNLP (Findings)*. 542–550.

[32] David Tweed and Andrew Calway. 2002. Tracking Multiple Animals in Wildlife Footage. In *ICPR*. 24.

[33] Dominique Verdejo and Eunika Mercier-Laurent. 2022. Video Intelligence as a component of a Global Security system. arXiv:2201.04349 [cs.AI]

[34] Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Sharon Li, and Neel Joshi. 2024. Is A Picture Worth A Thousand Words? Delving Into Spatial Reasoning for Vision Language Models. In *NeurIPS*.

[35] Renzhi Wu, Pramod Chunduri, Ali Payani, Xu Chu, Joy Arulraj, and Kexin Rong. 2024. SketchQL: Video Moment Querying with a Visual Query Interface. *Proc. ACM Manag. Data* 2, 4 (2024), 204:1–204:27.

[36] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *SIGMOD*. 602–616.

[37] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. CLEVRER: Collision Events for Video Representation and Reasoning. In *ICLR*.

[38] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*. 3911–3921.

[39] Enhao Zhang, Maureen Daum, Dong He, Manasi Ganti, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. 2023. EQUI-VOCAL Demonstration: Synthesizing Video Queries from User Interactions. *PVLDB* 16, 12 (2023), 3978–3981.

[40] Enhao Zhang, Maureen Daum, Dong He, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. 2023. EQUI-VOCAL: Synthesizing Queries for Compositional Video Events from Limited User Interactions. *PVLDB* 16, 11 (2023), 2714–2727.

[41] Enhao Zhang, Nicole Sullivan, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. 2025. Self-Enhancing Video Data Management System for Compositional Events with Large Language Models [Technical Report]. arXiv:2408.02243 [cs.DB]