

StreamingRAG: Real-time Contextual Retrieval and Generation Framework

Murugan Sankaradas
 murugs@nec-labs.com
 NEC Laboratories America
 Princeton, NJ, USA

Ravi K. Rajendran
 rarajendran@nec-labs.com
 NEC Laboratories America
 Princeton, NJ, USA

Srimat T. Chakradhar
 chak@nec-labs.com
 NEC Laboratories America
 Princeton, NJ, USA

ABSTRACT

Extracting real-time insights from multi-modal data streams from various domains such as healthcare, intelligent transportation, and satellite remote sensing remains a challenge. High computational demands and limited knowledge scope restrict the applicability of Multi-Modal Large Language Models (MM-LLMs) on these data streams. Traditional Retrieval-Augmented Generation (RAG) systems address knowledge limitations of these models, but suffer from slow preprocessing, making them unsuitable for real-time analysis. We propose StreamingRAG, a novel RAG framework designed for streaming data. StreamingRAG constructs evolving knowledge graphs capturing scene-object-entity relationships in real-time. The knowledge graph achieves temporal-aware scene representations using MM-LLMs and enables timely responses for specific events or user queries. StreamingRAG addresses limitations in existing methods, achieving significant improvements in real-time analysis (5-6x faster throughput), contextual accuracy (through a temporal knowledge graph), and reduced resource consumption (using lightweight models by 2-3x).

CCS CONCEPTS

- General and reference → Performance; • Information systems → Question answering.

KEYWORDS

Real-time Streaming systems, Retrieval-augmented Generation, Multi-modality, Knowledge Graphs, Video Understanding

ACM Reference Format:

Murugan Sankaradas, Ravi K. Rajendran, and Srimat T. Chakradhar. 2024. StreamingRAG: Real-time Contextual Retrieval and Generation Framework. In *Workshop on AI For Systems (AI4Sys '24), June 3–7, 2024, Pisa, Italy*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3660605.3660943>

1 INTRODUCTION

The ever-growing volume of streaming data presents great opportunities across diverse fields like healthcare for real-time analysis of medical images [6]; intelligent transportation systems for understanding traffic flow patterns [17]; and remote sensing, where satellite

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

AI4Sys '24, June 3–7, 2024, Pisa, Italy

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0652-3/24/06
<https://doi.org/10.1145/3660605.3660943>

constellations provide stream of imagery for applications like, environmental monitoring, economic activity measurement and disaster response [4, 14]. These domains share a common fundamental underlying need: the ability to extract real-time insights from video, image, and other multi-modal sensor streams. Dynamic situations and anomalies are frequent occurrences in these domains which need immediate attention. MM-LLMs [9, 13, 18] are often utilized due to their ability to efficiently extract information from multi-modal streams. Traditionally, the RAG framework [8] has been used to understand static data by gathering knowledge about the situation, which relies on fetching, indexing, and converting external data into structured formats - a time-consuming process. Hence it is unsuitable for real-time applications, leading to a potential gap in understanding critical events within the data stream. Extracting information using MM-LLMs requires significant amount of resources, which increases costs. For instance, MM-LLM can take up to 3-5 seconds per video frame in case of GPT-4V [12], which is impractical for real-time streaming applications as it can miss crucial unfolding events.

To address limitations of existing methods and achieve real-time comprehension, we propose StreamingRAG, a framework that utilizes efficient models to construct an evolving knowledge graph [7] about the stream content. StreamingRAG accomplishes this by extracting contextual scene-object-entity relationships. This knowledge graph serves two key purposes: (1) providing an efficient scene representation and (2) facilitates real-time and contextual information retrieval. Extracting all actors and their relationships in a real-time scene is computationally expensive.

Therefore, our research proposes a dynamic priority-based approach for knowledge extraction. We prioritize information about specific actors and relationships based on user constraints, evolving events, and current context. The scheduler orchestrates this dynamic prioritization and utilizes a multi-modal question generator to ask relevant questions at a given time. This approach ensures that key information about prioritized actors and relationships is extracted within the allotted time frame, ultimately leading to the construction of knowledge graph in real-time that reflects the most crucial aspects. We make the following key contributions.

- StreamingRAG, a Retrieval-Augmented Generation (RAG) system for streaming data, tackles the challenge of querying real-time data streams, by constructing scene-aware spatio-temporal knowledge graphs using lightweight models.
- Context-driven dynamic priority-based knowledge extraction process, which enables the efficient construction of knowledge graph.
- We evaluate the effectiveness of the StreamingRAG in monitoring video streams for Intelligent Transportation System. It yields significant advantages: 5-6x faster processing (throughput) compared to prior work, ensuring real-time analysis. It

maintains contextual accuracy in evolving scenarios using temporal knowledge graph. Finally, the use of lightweight models reduces resource utilization by 2-3x.

2 MOTIVATION

Traditional RAG systems enhance the capabilities of MM-LLMs by supplementing their knowledge with retrieved external data sources. Using semantic search, retriever module builds enriched context by fetching relevant information from external sources, which is then incorporated to enrich query context so that LLM can answer questions, create descriptions, and complete tasks by combining the multi-modal input and retrieved relevant data. Existing systems rely on computationally expensive MM-LLMs to process data and store the extracted information. The effectiveness of this approach is measured by three factors: information loss due to processing time, response delay, and computational resources required.

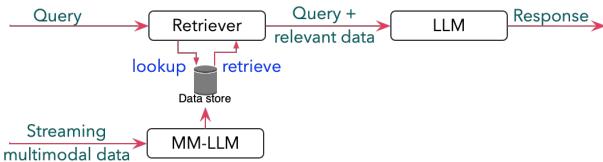


Figure 1: Baseline architecture

2.1 Limitations of baseline architecture

Heavyweight MM-LLMs have lengthy processing times, which limits their use to batch processing systems, where offline processing is done. This approach is not suited for realtime streaming scenarios where immediate analysis is needed on streaming data as it arrives. Key metrics of streaming applications are as follows.

Accuracy: Baseline fails to generate accurate responses for an unfolding event, missing crucial details and failing to adapt evolving context. Inaccurate timing and sequencing caused by missing to process enough data chunks, disrupt the order of events, creating time gaps and potentially misinterpreting the flow of the stream. Anomaly detection becomes challenging as crucial deviations from the norm might be masked by missing data.

Latency & Throughput: Powerful heavyweight models demand significant resources, leading to large execution times, leading to reduced throughput and responsiveness. Furthermore, obtaining spatial details necessitate generative models to produce more elaborate responses, potentially exceeding over 50 tokens. For instance, for a vision-to-language model, the latency associated with various output response token sizes of the ShareGPT4V VLM [2] is shown in Figure 2. Evidently, to function in real-time, the output token sizes must be minimized, reinforcing the importance of employing tailored lightweight models to address targeted questions pertaining to descriptive inquiries.

Cost: MM-LLMs have immense computational demands which require state of the art hardware accelerators. Processing hundreds of sensor feeds concurrently for deployments like smart city applications in real-time can quickly overwhelm available hardware. Increased latency hinders the ability of the system to provide timely insights,

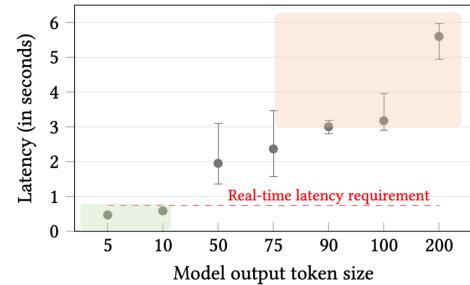


Figure 2: Latency vs output token size of ShareGPT4V [2]. Red shaded area indicates descriptive answers, using larger output token size which increases latency. Green shared area generates less descriptive output meeting real-time constraints using smaller output token size. Need to ask descriptive questions to model so that spatio-temporal information across frames is extracted.

while reduced contextual accuracy compromises the quality and usefulness of generated responses.

2.2 Our approach

Instead of using heavy duty models to extract information from real-time streaming content, StreamingRAG tackles the challenge, as shown in Figure 3. Unlike traditional methods burdened by heavyweight models, StreamingRAG prioritizes both temporal context and efficiency by systematic extraction and dynamic knowledge pipelines, as explained in Section 3. This is achieved through extracting low-level object information and their relationships instead of the high-level scene by dynamically prioritizing information about specific entity-attribute-relationship and creating embeddings for incoming streams followed by the construction of knowledge graphs. StreamingRAG maintains contextual accuracy in evolving scenarios by utilizing the temporal knowledge graph. By prioritizing efficiency and context-awareness, StreamingRAG enables efficient exploration of real-time data streams.

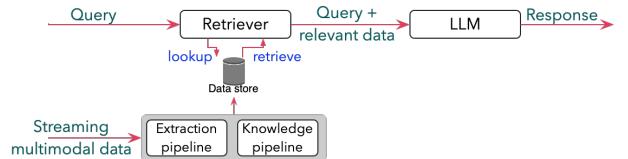


Figure 3: StreamingRAG approach

3 STREAMING RAG

3.1 Overview

StreamingRAG framework handles both real-time continuous monitoring through standing queries and interactive queries. Standing queries (aka persistent queries), are ongoing queries that continuously scan real-time streams for specific updates/conditions. They are different from one-time queries with a finite response, offering a continual awareness of specific events or patterns within the data.

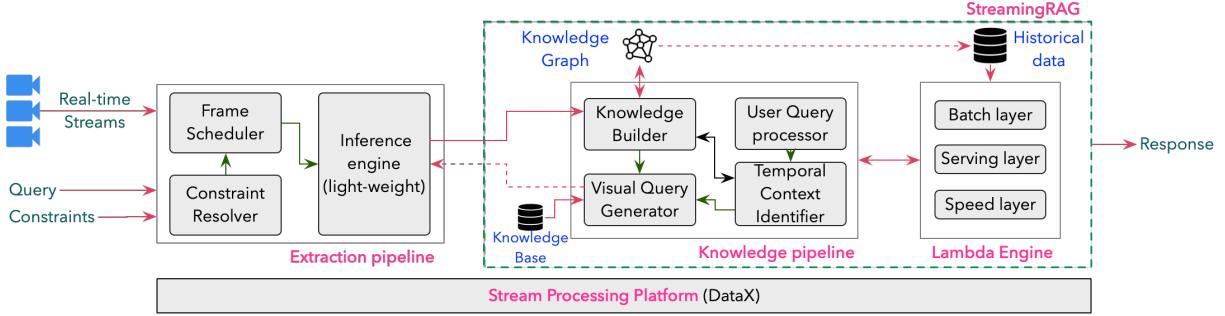


Figure 4: StreamingRAG: System Architecture

Interactive queries, which are submitted by users on demand, require both real-time and historical data. They allow users to refine their query based on the initial response, then dive deeper into specific aspects. StreamingRAG (Figure 4) consists of 4 main components: (a) extraction pipeline, (b) knowledge pipeline, (c) stream processing platform, and (d) lambda engine as presented below.

3.2 Extraction Pipeline

Spatial metadata is extracted using various inference engines (i.e., VLMs, LLMs, and MM-LLMs etc.) from frames, subject to real-time constraints. Data selection from input streams is prioritized dynamically based on evolving events, due to impracticality of processing every incoming data chunk.

3.2.1 Frame Scheduler: Frames are received from streaming video sources to input stage of the extraction pipeline, which is running on top of the stream processing platform (3.4). These frames are then directed to models of different capabilities (3.2.3). The scheduler analyzes incoming frames to assess their content and complexity, considering factors like motion and scene details. System-level constraints are simultaneously monitored by a Constraint Resolver (3.2.2). This combined analysis allows the frame scheduler, with both prioritized rules and adaptive algorithms, to select the optimal frame rate of input data being fed to the inference engines.

3.2.2 Constraint Resolver: The constraint resolver optimizes the dynamic scheduling of frames across the models within the real-time extraction pipeline to enhance system responsiveness. Factors like user-specified constraints (frames per second, model processing latency, inference cost) and system-level operational constraints (CPU usage, GPU load, memory availability) are optimized to maximize throughput while ensuring latency constraints.

3.2.3 Inference Engine: The real-time extraction pipeline uses a combination of heavyweight and lightweight Multi-Modal Large Language Models. The models take in frames and list of queries to be processed on the given frame. The VLMs are specifically dedicated to handling questions from the question bank for a tiered analysis, acting as a filter for any event detection by optimizing computational resources.

3.3 Knowledge Pipeline

While extraction pipeline extracts only the metadata from input streams, knowledge pipeline translates the metadata into actionable knowledge, enabling the system to answer user queries and, more importantly, provide feedback to the extraction pipeline, guiding its metadata extraction from incoming streams. These include analyzing the current response, constructing temporal context using spatial details across frames, monitoring events as they unfold, refining queries for subsequent frames. The functional components in knowledge pipeline are described below.

3.3.1 Knowledge Base: Knowledge Base (*KB*) serves a foundational repository of common-sense knowledge, providing an initial context for understanding incoming streaming data. It aids in Knowledge Graph initialization [1, 11, 15, 16], forming a starting point for dynamic context building. *KB* depends on the use cases. For instance, in the traffic scenario, *KB* contains information such as different actors like pedestrians, people, drivers, objects like vehicles and their relationships, traffic rules, and historical traffic patterns.

3.3.2 Knowledge Graph: A knowledge graph (*KG*) is represented using semantic tuple of three elements: subject, predicate, and object. The subject and object represent an entity pair, such as a person-location, or an object-property. The predicate specifies the relationship connecting these entities. *KG* acts as the dynamic memory, capturing the contextual details necessary for response generation. The information within the *KG* is contingent upon the context of the specific use case, and it can be sourced from the *KB*.

3.3.3 Knowledge Builder: Knowledge Builder updates and refines the *KG* in real-time based on *KB* and user-directed contexts. It assembles entities, relationships, and contextually relevant details from the *KB* into the evolving *KG*. It continuously updates the graph, reflecting evolving context, and adapts to influx of streaming data, ensuring that the graph remains reflective of the real-time event, allowing downstream components, such as the retriever and language models, to operate on up-to-date context. Generating *KG* from the scene-level spatial understanding is achieved by modeling $Pr(G|F) = Pr(B, L, R)$, where F is the input frame, G is the desired graph, $b_i \in B$ is the bounding box in the images, L is the object labels and R is the relations among the objects L .

3.3.4 Temporal Context Identifier: Understanding incoming streams requires establishing temporal context (e.g., event or incident). This

context acts as a trigger to schedule resource allocation, such as processing power (CPU/GPU) or higher frame rates, ensuring critical details aren't missed during crucial moments. Combining spatial information extracted from each frame in the extraction pipeline provides a comprehensive understanding of the temporal context. Extracting temporal context involves iteratively following these steps: Identify the user query's specific needs; Fetch relevant records from the data store; Prioritize retrieved content through ranking and filtering via moderation. Identifying context and extracting necessary information from incoming frames involves querying the knowledge base (KB) to identify associated entities and relationships. Entities with the highest probabilities are selected, prompting the construction of prompts for subsequent iteration of questions within the VLM inference engine in the extraction pipeline. Once an event concludes or under steady-state conditions, the KG undergoes reset, ensuring alignment with the current state and to prepare for subsequent events and queries.

3.3.5 Visual Query Generator: VQG is an interface between extraction and knowledge pipelines, incorporating the current video response to pull pertinent information from KG. Given the current video response S_t and questions Q_t , it interfaces with KG through the temporal context identifier at the time instant t . Query generator integrates S_t with the KB^t to update the set of questions, $Q_{t+k} \leftarrow VQG(S_t, KB^t)$ where $k > 0$ based on the temporal cues. This process refines the queries, incorporating the evolving context. The refined questions, $q \in Q_{t+k}$, are then scheduled for the subsequent frames f_{t+k} are then embedded into the current prompt P , which is dispatched to the extraction pipeline using the scheduler.

3.3.6 User Query Processor: Query processor translates the unstructured, multi-modal user queries into actionable executors for the pipeline. In addition to parsing and interpreting user queries, processor refines queries based on the evolving context and KB .

3.4 Stream Processing Platform

The system components are hosted on real-time stream processing platform DataX [3]. It tackles the complexity of building distributed stream processing applications by streamlining data exchange, transformations, and fusion. DataX provides an abstraction, which simplifies application specification by exposing parallelism and dependencies among microservices. This allows the DataX runtime to automatically manage data communication and provides dynamic scaling and optimization.

3.5 Lambda Engine

Lambda engine is used to support both real-time standing queries and interactive queries. The framework requires access to both real-time and historical data, and this access is expedited to minimize query response time through three layers: batch, serving and speed layer. The batch layer pre-processes and updates KG with historical context, providing foundation for contextual understanding. This result is then infused to the real-time extraction and knowledge pipelines. The serving layer enables efficient retrieval of relevant data from KG , enhancing the retrieval process for user's interactive queries. The speed layer handles real-time data streams, so that system is adaptive and responsive to evolving contexts.

4 EXPERIMENTS & EVALUATIONS

4.1 Setup

StreamingRAG is evaluated in the context of Intelligent Transportation Systems (ITS) for anomaly event detection by generating event descriptions, using both public [19] and proprietary dataset, for various event complexities, with a specific focus on evaluating real-time evolving events through following standing queries. (1) *Raise an alert when there is an accident involving collision between vehicles and people.* (2) *Inform me upon spotting pedestrians in distress.*

Our experimental setup consists of a cluster of servers, each equipped with a 32-core Intel Xeon CPU, 256 GB of memory, and 24 GB NVIDIA GeForce RTX 3090 Ti. Our evaluation methodology involves streaming the videos captured by an IP camera.

Extraction pipeline utilizes ShareGPTV4 [2], a descriptive image captioning model, for inference in our baseline system. VQA models, BLIP [10] and BLIP-2[9], are used to extract spatial information.

Streaming Processing Platform is built using DataX [3] real-time stream processing framework. It enables us to efficiently manage the continuous data flow from camera feeds, process it using the chosen VLM models, and integrate it with knowledge pipeline.

Inside *Knowledge Pipeline*, KB is manually created for our traffic monitoring use case, currently implemented as a mapping between entities, potential attributes and relationships among them. KG is maintained using the NetworkX [5]. Upon receiving responses from the VLM for the previous set of VQG generated queries, the generator retrieves relevant entities, attributes and necessary relationships between entities, from KB . Subsequently, it formulates a prompt comprising a set of upto five questions, depending on the scenario context, to stay within the real-time latency requirement.

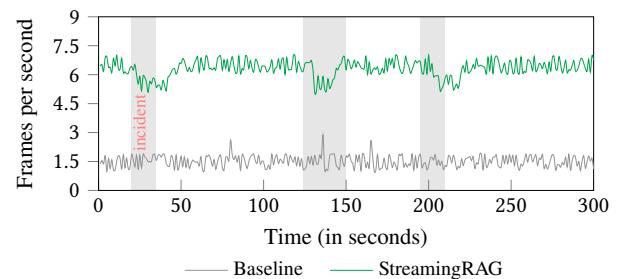


Figure 5: Throughput

4.2 Evaluation Metrics

Our evaluation considers both the duration of the event and its complexity. Duration of an event is critical for the real-time analysis because if a frame takes a longer time to process, a short-time event will be completely missed. Long-time events are more manageable in the real-time extraction phase, due to system's ability to observe and formulate questions for subsequent frames. System performance is evaluated using the following metrics.

Latency & Throughput – Time taken to generate responses.

Cost (GPU resource utilization) is measured in terms of GPU memory usage, a factor determining the scalability of the system.

Accuracy of generated responses is evaluated through human examination, with a specific focus on system's adeptness at identifying evolving contexts and creating responses. In our use case, evaluation is done on two factors (a) context-aware event detection (b) response generation. For instance, in hit-and-run, the system needs to identify the exact sequence: vehicle-person collision, immobility of person following the collision and vehicle fleeing the scene. Missing any event loses the context, leading to an incorrect response. Generation phase assesses the accuracy and adequacy of model's responses (both LLMs and VLMs).

4.3 Results

4.3.1 Latency & Throughput: The shaded areas in the graph (Figure 5) denote instances of anomaly incidents such as hit-and-run and vehicle collisions. In baseline, higher inference times causes crucial temporal information loss when frame-level responses are aggregated to create response. However, StreamingRAG operates at approximately 8 fps (1/3rd of video's fps), effectively leading to extract contextual spatial information.

4.3.2 GPU resource utilization: While Table 1 highlights the efficiency gains of StreamingRAG for a single camera stream with a dedicated GPU, real-world deployments involve hundreds of cameras distributed across a wide area. Our architecture allows GPU sharing, to dynamically adjust resource allocation based on the evolving context of each stream, ensuring efficient resource utilization even in complex and geographically distributed deployments.

Table 1: GPU resource utilization

	Baseline	With StreamingRAG
VLM	18GB	8GB
LLM Summarizer	-	2GB
Total	18GB	10GB

4.3.3 Accuracy: In real-time scenarios, the ability to promptly detect unfolding incidents is critical. We achieved a notably higher detection rate as shown in Table 2, due to its faster processing capabilities and the utilization of historical frame-level responses. Despite this improvement, both systems encountered challenges in promptly detecting short-term events, although ours demonstrated a relatively higher success rate compared to baseline. Generated responses are evaluated based on accuracy of the chronological sequence of events accumulated temporally. Table 3 shows the responses of 4 events such as hit-and-run, vehicle-to-vehicle collision, vehicle-to-pedestrian collision and people commotion. The baseline struggles to process frames promptly, resulting in a failure to gather information about the individuals and vehicles involved, ultimately missing the collision events entirely. For instance, in Vehicle-to-Vehicle collision, the baseline correctly identified that a car was on fire. However, it failed to capture the context of the collision between vehicles, which resulted in the car catching on fire. In vehicle-to-pedestrian collisions, the baseline not only failed to detect the collision event but also misclassified the situation, interpreting it as a game rather than within the context of a traffic scenario. With StreamingRAG, in hit-and-run case, the exact moment of

Table 2: Results of event detection on various event types

Type	Anomaly Event			Events detected	
	No. of Videos	Average Duration (in seconds)	Ground truth (number of events)	Baseline	StreamingRAG
Hit and run	2	8	3	0	2
Vehicle-to-Vehicle collision	2	16	3	1	3
Vehicle-to-Pedestrian collision	1	48	1	0	1
People commotion	1	26	1	0	1
			Total events: 8	1 (12.5%)	7 (87.5%)

the accident was overlooked. However, through analyzing multiple frames, it found the person was lying on the road near the car, indicating the occurrence of an accident (highlighted in red).

Table 3: Accuracy evaluation of baseline and StreamingRAG based on context-aware response generation

Event	Generated responses	
	Baseline	StreamingRAG
Hit and run (8s)	The image captures a winter scene on a city street. Two individuals are seen walking on the sidewalk, bundled up against the cold. A car is parked on the street, its surface dusted with snow. The street itself is lined with trees, their branches bare, a common sight in winter. The sky overhead is a blanket of gray, hinting at the possibility of more snowfall. The overall atmosphere is quiet and serene, a typical winter's day in a city.	The scene has good daylight and the background consists of a woman in a red coat and red hat standing on the corner of a city street. There are trucks present as vehicles. In the image, a woman in red pants and a red jacket is seen standing on the street corner. <i>There is a person lying down on the street near a car, indicating a potential accident</i>
Vehicle-to-Vehicle collision (11s)	<i>A car is on fire</i>	The scene depicts a highway at night. People in the scene are observed driving on the highway, where cars and trucks are present. Notably, <i>two cars adjacent to each other, with one of them engulfed in flames within the image, suggesting a potential collision between the vehicles</i> .
Vehicle-to-Pedestrian collision (48s)	The image captures a scene of urban decay, dominated by a pile of rubble and debris. The rubble, a chaotic mix of bricks, concrete, and other construction materials, is scattered haphazardly across the street. Amidst this disarray, a yellow sign stands out, bearing the words "Games for Windows" and "Games for Windows Live". The sign, once a beacon of digital entertainment, now seems out of place amidst the physical detritus.	The scene depicts a dark daylight setting with a background consisting of a pile of garbage. Various types of vehicles, particularly those used for food transportation, are present in the area. <i>There may be a potential crash because the image shows a chaotic scene with rubble scattered around and a car crushed by the debris. People in the scene are observed attempting to save their lives amidst the chaotic environment</i> .
People commotion (26s)	The image captures a scene in a parking lot. A police officer, dressed in a black uniform, stands in the center of the frame, holding a walkie-talkie. The officer is positioned in front of a silver SUV, which is parked in a handicapped spot. The parking lot is filled with several other vehicles, including a black van and a red truck.	The scene features a variety of vehicles present in a parking lot environment. The background consists of this parking lot, <i>where a man walking his dog was struck by a car</i> . The scene is illuminated with good daylight. In the image, a car is depicted within the parking lot alongside individuals on the ground and <i>some engaged in a physical altercation</i> . Additionally, <i>police presence is noted within the scene</i> .

5 CONCLUSION

In this paper, we addressed the limitations of high processing demands and latency associated with using Multi-Modal Large Language Models for real-time information extraction from streaming data. We proposed StreamingRAG, a framework that leverages lightweight models to construct an evolving knowledge graph capturing scene-object-entity relationships. Our results demonstrate significant improvements in terms of contextual temporal accuracy, reducing the resource utilization and costs. Additionally, StreamingRAG facilitates interactive querying, which enables real-time decision-making in critical scenarios.

REFERENCES

- [1] Sören Auer, Christian Bizer, Jakob Koblenz, Jens Lehmann, Sebastian Hellmann, Illya Issa, and Ronald Arndt. 2007. DBpedia - A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *The Semantic Web Journal* 8, 1 (2007), 77–114.

- [2] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2023. ShareGPT4V: Improving Large Multi-Modal Models with Better Captions. arXiv:2311.12793 [cs.CV]
- [3] Giuseppe Coviello, Kunal Rao, Murugan Sankaradas, and Srimat Chakradhar. 2022. DataX: A System for Data eXchange and Transformation of Streams. In *Intelligent Distributed Computing XIV*, David Camacho, Domenico Rosaci, Giuseppe M. L. Sarné, and Mario Versaci (Eds.). Springer International Publishing, Cham, 319–329.
- [4] Mikolaj Czerkawski, Robert Atkinson, and Christos Tachtatzis. 2023. Detecting Cloud Presence in Satellite Images Using the RGB-based CLIP Vision-Language Model. arXiv:2308.00541 [cs.CV]
- [5] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
- [6] Iryna Hartscock and Ghulam Rasool. 2024. Vision-Language Models for Medical Report Generation and Visual Question Answering: A Review. arXiv:2403.02469 [cs.CV]
- [7] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D’amatto, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *Comput. Surveys* 54, 4 (July 2021), 1–37. <https://doi.org/10.1145/3447772>
- [8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL]
- [9] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. arXiv:2301.12597 [cs.CV]
- [10] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. arXiv:2201.12086 [cs.CV]
- [11] Google LLC. 2022. Google Knowledge Graph Search API. <https://developers.google.com/knowledge-graph>
- [12] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
- [13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. <https://proceedings.mlr.press/v139/radford21a.html>
- [14] Jonathan Roberts, Kai Han, and Samuel Albanie. 2023. SATIN: A Multi-Task Metadataset for Classifying Satellite Imagery using Vision-Language Models. arXiv:2304.11619 [cs.CV]
- [15] Amazon Science. 2021. Combining Knowledge Graphs, Quickly and Accurately. *Amazon Science Blog* (2021). <https://www.amazon.science/blog/combining-knowledge-graphs-quickly-and-accurately>
- [16] GeoNames Team. 2005. GeoNames Geographical Database. <https://www.geonames.org/>
- [17] Matthew Veres and Medhat Moussa. 2020. Deep Learning for Intelligent Transportation Systems: A Survey of Emerging Trends. *IEEE Transactions on Intelligent Transportation Systems* 21, 8 (2020), 3152–3168. <https://doi.org/10.1109/TITS.2019.2929020>
- [18] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and Tell: A Neural Image Caption Generator. arXiv:1411.4555 [cs.CV]
- [19] Papers with Code. 2024. Abnormal event-detection in Video. <https://paperswithcode.com/task/abnormal-event-detection-in-video>