Graph-Enhanced Retrieval Framework for Event-Based Image Retrieval

Ayush Saun
MT24024
Advanced Deep Learning Project
EVENTA 2025 Grand Challenge - Track 2

October 28, 2025

1 Introduction: What's the Problem?

1.1 The Core Challenge

Imagine you're trying to find a specific photo from a news event, but all you have is a text description like *"massive protests following disputed election results in Argentina."*

Why is this hard?

Normal image search works great for simple things like "a dog playing fetch" where the words directly match what you see in the image. But for complex events:

- **Event descriptions are long and complex** - Often 100+ words describing who did what, when, why, and where. Models like CLIP only handle 77 tokens (about 50-60 words).
- **Words don't directly match visuals** - Terms like "disputed," "controversial," or "crisis" don't have obvious visual elements. You can't just look for objects.
- **Events span multiple articles** - One event gets covered in many articles published over days or weeks. Each article shows different angles of the same story.
- **Images aren't perfectly matched** - News articles have 1-30 images, and they're not always perfectly aligned with what the caption describes. Some images are tangential or show context rather than the main event.

1.2 My Original Proposal

According to my ADL project proposal titled *"EventVision: Context-Aware Image Retrieval from Informative Caption,"* **I wanted to develop an advanced retrieval framework that can handle long, complex event descriptions by better utilizing the rich contextual information available in event captions.**

The key insight from my proposal is that event captions contain way more information than object-level descriptions, and we need to build systems that can actually use all that context instead of just matching keywords.

1.3 The EVENTA 2025 Challenge

I'm working on **Track 2: Event-Based Image Retrieval (Closed Track)**.

The task is straightforward:

- **Input**: A text caption describing a real-world event
- **Output**: The single best matching image from a database of 415,000+ images
- **Constraint**: Can only use the provided dataset and publicly available pretrained models (no external data)

How performance is measured:

- Mean Average Precision (mAP)
- Mean Reciprocal Rank (MRR)
- Recall@1, Recall@5, Recall@10
- Overall Score (weighted combination of all metrics)

2 The Dataset: OpenEvents V1

2.1 What's in the Dataset

The OpenEvents V1 dataset was specifically created for this challenge. It's a massive collection of real news articles and their images.

Component	Count
Total Articles	**202,803**
From CNN (2011-2022)	24,200
From The Guardian (2019-2025)	178,603
Total Images	**415,324**
Average Images per Article	2.05
Training Queries	20,000+
Validation Queries	2,000+
Test Queries	5,000+

Table 1: OpenEvents V1 Dataset Statistics

2.2 Why This Dataset is Special

Unlike typical image-caption datasets like MS-COCO where captions just describe what's visible ("a person riding a bicycle"), OpenEvents is different:

1. **Event-Centric Captions**: Descriptions focus on events and narratives, not just objects
    - Example: "Opposition parties condemn government's response to economic crisis" vs. "People at podiums"

- The caption talks about abstract concepts (condemn, crisis, response) that aren't directly visible

2. **Long, Complex Narratives**: Average caption is 100-200 tokens
   - Includes temporal context ("following weeks of tensions...")
   - Has causal relationships ("sparked by controversial ruling...")
   - Mentions multiple entities (people, places, organizations)
3. **Real News Context**: Each image comes from an actual news article with full text
   - Articles provide rich background information
   - Images are embedded in journalistic context
   - Temporal information (publication dates) is available
4. **Weak Image-Caption Alignment**: Images aren't perfectly matched to captions
   - One article has multiple images showing different aspects
   - Captions describe events abstractly; images show specific moments
   - This is realistic but makes retrieval harder

3 What I Learned from Reading Papers

I read 7 papers to understand how people are solving this problem and what gaps exist. Let me walk through each one and explain what they did, what worked, and **what was missing**.

3.1 Paper 1: OpenEvents V1 Dataset Paper

What it introduced:

This paper introduced the dataset itself - 202,803 articles and 415,324 images from CNN and The Guardian. It defined three tasks:

1. Event-enriched image captioning (generate captions for images)
2. Event-based article retrieval (find relevant articles)
3. Event-based image retrieval (find relevant images) - **this is my focus**

Key learning:

The paper showed that **because images are tied to articles, a two-stage approach makes sense: caption → article → image**. First find relevant articles, then select images from those articles. This is smarter than trying to directly match captions to 415K images.

What I took away:

The dataset is huge, so whatever approach I build needs to be efficient. The article-centric organization suggests that understanding article relationships could be valuable.

3.2 Paper 2: EVENT-Retriever (Winning Solution, Score: 0.5766)

What they built:

This is the best-performing system. They used a four-stage pipeline:

1. **Stage 1 - Dense Article Retrieval:**
    - Used Qwen3-Embedding to convert all 202K articles into vectors (1024 dimensions each)
    - Stored these vectors in a database (FAISS) for fast searching
    - When a query comes in, embed it with the same model
    - Find top-100 most similar articles by computing cosine similarity
    - This is like finding needles in a haystack but using smart math
2. **Stage 2 - LLM Reranking:**
    - Take those 100 articles and use a large language model (Qwen3-Reranker) to rerank them
    - The LLM reads each article and answers: "Does this article help identify the image for this caption?"
    - Extract the probability of "yes" as the relevance score
    - Keep only top-10 articles after reranking
    - **This is powerful because LLMs can understand context and reasoning**
3. **Stage 3 - Image Collection and Scoring:**
    - From the top-10 articles, collect images (at least 10 images from at least 3 different articles)
    - Score each image using gme-Qwen2-VL (a vision-language model)
    - This model can handle long captions (unlike CLIP which maxes out at 77 tokens)
    - Compute similarity between caption and each image
4. **Stage 4 - Reciprocal Rank Fusion:**
    - Run the entire pipeline multiple times with different settings (different model sizes, different parameters)
    - Combine all the results using a smart fusion method
    - Images that appear at top positions across multiple runs get boosted
    - This ensemble approach improves robustness

Their results:

- Achieved **0.5766 overall score** (best in competition)
- Recall@1: 0.469 (correct image in top-1 position 47% of the time)

- Recall@10: 0.744 (correct image in top-10 positions 74% of the time)

What was good:

They used a zero-shot approach (no training on OpenEvents), just pretrained models. The hierarchical pipeline (caption → article → image) worked really well. LLM reranking added significant value.

What was missing:

Here's the problem: Articles are retrieved independently based only on similarity between the caption and each article. There's no modeling of relationships between articles.

Let me explain why this matters with an example:

Example scenario:

Query: "Massive protests erupt after controversial court ruling"

- Article A: "Thousands take to streets following Supreme Court decision" - Similarity: 0.89 Retrieved
- Article B: "Police deploy tear gas to disperse demonstrators" - Similarity: 0.72 × Missed
- Article C: "Opposition leaders call for peaceful demonstrations" - Similarity: 0.68 × Missed

Even though B and C are clearly part of the same event (published hours apart, share entities like protest location, same event), they get missed because their direct similarity to the caption is lower. **But Article B might have the most dramatic image showing the confrontation!**

3.3 Paper 3: Hierarchical Article-to-Image (Rank 3, Score: 0.563)

What they did differently:

Instead of using just one text representation, they used **four different strategies** to represent article text:

1. **Sparse Text (BM42):**
   - Keyword-based matching (like old-school search engines)
   - Good for exact matches ("election" in query matches "election" in article)
   - Fast but misses semantic similarity
2. **Chunked Text (Jina v3):**
   - Split long articles into 512-token chunks
   - Embed each chunk separately
   - Score article based on best-matching chunk
   - Handles long articles better than processing all at once
3. **Long-Context Text (Jina v2 with Late Chunking):**
   - Process entire article first to get document-level context

- Then chunk for efficiency
- Each chunk "knows" about the whole document
- Better than independent chunking

4. **Document-Level (ColQwen2.5):**
    - Treat article PDFs as images!
    - Use vision-language model to "see" the document layout
    - Captures visual structure (headers, formatting, etc.)

After retrieving with all four methods, they combined results and applied reranking.

What worked:

Using multiple text representations improved robustness. Different strategies catch different things - sparse catches exact keywords, dense catches semantics, document-level catches structure.

What was still missing:

Same problem as Paper 2 - **articles are treated independently, no relationship modeling.**

3.4 Paper 4: Hybrid Dense-Sparse Framework (Rank 2, Score: 0.572)

Their key insight:

They identified a critical problem: **CLIP only handles 77 tokens, but news articles have 14,000+ tokens on average.** This causes massive information loss.

Their solution:

1. **Separate text and image processing** (don't force them into the same short-token model)
2. **Use semantic chunking for text:**
    - Instead of arbitrary token splits, chunk based on semantic boundaries (paragraphs, sections)
    - 384 tokens per chunk with 64-token overlap (so context isn't lost at boundaries)
    - Use Nomic-embed-text-v1.5 which handles 8,192 tokens
3. **Combine dense + sparse retrieval:**
    - Dense (Nomic embeddings): Captures semantic meaning
    - Sparse (BM25): Captures exact keyword matches
    - Fuse with Reciprocal Rank Fusion

Their amazing finding:

Proper semantic chunking alone improved performance from 0.17 to 0.52 - a 35-point gain! This shows how critical text preprocessing is.

What was still missing:

Yet again - **no modeling of article relationships.**

3.5 Paper 5: HGAN (Hierarchical Graph Alignment Network)

What they proposed:

A graph-based approach for image-text retrieval. This is where I got interested in graphs.

Their architecture:

1. **Feature Extraction (Frozen):**
    - ResNet-152 for global image features
    - Faster R-CNN for local image regions (object detection)
    - BERT for text features
    - These encoders stayed frozen (not trained)
2. **Graph Construction:**
    - Built graphs where nodes are image regions or text words
    - Edges represent relationships between these features
    - For example: "dog" (text node) connects to dog-region (image node)
3. **Processing (Trainable):**
    - **Important clarification:** Despite the name, HGAN doesn't actually use traditional Graph Convolutional Networks from PyTorch Geometric
    - Instead, it uses attention mechanisms (multi-head attention + GRU)
    - Only the alignment modules (MFA and MFAR) are trained

What I learned:

Graph structures CAN help with multimodal alignment. **But HGAN works at the feature level (within single image-text pairs), not at the document level (between different articles).**

Key difference from what I want to do:

- HGAN: Builds graphs of image regions and text words within one image-text pair
- My approach: Build graphs of entire articles across the whole corpus

3.6 Paper 6: ImageBind

Core idea:

Images can act as a "universal connector" between different modalities.

How it works:

1. Train Model 1 on image-text pairs
2. Train Model 2 on image-audio pairs
3. **Magic:** Without ever training on text-audio pairs, the models can now match text to audio!

Why this works:

Image features are so rich that when you align text to images and audio to images, text and audio automatically align to each other through the image space.

What I learned:

Joint embedding spaces are powerful. But this doesn't help with my specific problem of modeling article relationships.

3.7 Paper 7: Multimodal GNN Framework

What they proposed:

Using Graph Neural Networks for multimodal information retrieval. **This is where I found the core concept for my approach.**

Their approach:

1. **Build a graph:**
   - Nodes = features from different modalities (text, image, etc.)
   - **Intra-modal edges** = connections within the same modality (e.g., article ↔ article)
   - **Inter-modal edges** = connections across modalities (e.g., text ↔ image)
2. **Use GCN or GAT:**
   - Graph Convolutional Networks (GCN) or Graph Attention Networks (GAT)
   - These do "message passing" - information flows through the graph
   - Each node aggregates information from its neighbors
3. **Add cross-modal attention:**
   - Dynamically weight different modalities based on the query
   - Some queries need more text, others need more visual info

What I learned:

This is the key insight I'm building on: GNNs can model relationships in multimodal data through graph structure and message passing.

What was missing:

- Only tested on small medical datasets (few thousand samples)
- **Never applied at the scale of 200K+ articles**
- No pretrained models available
- Not applied to event-based retrieval specifically

4 The Gap: What's Missing in All These Approaches

4.1 The Core Problem Nobody Addressed

After reading all these papers, I found one consistent gap:

All top-performing methods retrieve articles independently based only on pairwise similarity between the query and each article. They completely ignore relationships between articles.

4.2 Why This Matters - A Detailed Example

Let me show you exactly why ignoring article relationships is a problem.

Query Caption: "Massive protests erupt after controversial court ruling"

What current systems do:

1. Embed the caption: [vector of 1024 numbers]
2. Compute similarity with each of 202,803 articles independently
3. Rank articles by similarity score

Results from current approach:

Article	Similarity	Retrieved?
A: "Thousands take to streets following Supreme Court decision"	0.89	Yes
B: "Police deploy tear gas to disperse demonstrators"	0.72	× No
C: "Opposition leaders call for peaceful demonstrations"	0.68	× No
D: "Analysis: Legal implications of court ruling"	0.65	× No

Table 2: Current Approach - Independent Similarity

Why articles B, C, D got missed:

- Article B focuses on police action (not explicitly in caption)
- Article C focuses on opposition response (one step removed from query)
- Article D is analytical/contextual (different writing style)

But here's the thing:
All four articles are about the SAME EVENT!

- Published within 6 hours of each other
- All mention the same court ruling
- Share entities: location (same city), court name, case details

- Article B shows the most dramatic images (police confrontation)
- Article C has great images of opposition leaders speaking

Current systems only retrieve Article A and miss the potentially better images in B, C, and D.

4.3 What My Graph Approach Would Do

With graph structure:

Connection	Why Connected
A ↔ B	Temporal: 3 hours apart; Entity: same location, "protest", "police"
A ↔ C	Temporal: same day; Entity: "opposition", same event
A ↔ D	Semantic: 0.78 similarity; Entity: same court case
B ↔ C	Temporal: 2 hours apart; Entity: "demonstrators", same location

Table 3: Graph Connections Between Articles

How GNN retrieval works:

1. Initial retrieval finds Article A (high similarity 0.89)
2. GNN does message passing through the graph
3. Information propagates: A → B, A → C, A → D
4. B, C, D get boosted because they're connected to A
5. Now all four articles are in the candidate set
6. We consider images from all four articles instead of just A

Result: We find better images because we explored the event context through graph structure.

5 My Proposed Solution: Graph-Enhanced Retrieval

5.1 The Core Concept

Main idea: Instead of treating articles as independent entities, explicitly model their relationships through a graph structure. Then use Graph Neural Networks to explore these connections during retrieval.

5.2 What I Mean by "Graph"

Think of it like a social network, but for articles:

- **Nodes**: Each of the 202,803 articles is a node
- **Edges**: Connections between articles based on three types of relationships

Three types of edges:

1. **Temporal Edges:**
    - Connect articles published close in time
    - Example: Article from March 15 connects to article from March 17
    - Logic: Events unfold over days/weeks; related articles cluster temporally
    - Implementation: Connect articles within 14-day window
2. **Entity Edges:**
    - Connect articles that share named entities (people, places, organizations)
    - Example: Two articles both mention "Argentina" and "Supreme Court"
    - Logic: Articles about the same entities likely cover related events
    - Implementation: Use spaCy to extract entities, connect if more than 30% overlap
3. **Semantic Edges:**
    - Connect articles with high embedding similarity
    - Example: Two articles about economic crises in different countries (similar themes)
    - Logic: Semantically similar articles might provide relevant context
    - Implementation: Connect if cosine similarity greater than 0.7

5.3 How It's Different From HGAN

People might confuse my approach with HGAN since both use "graphs," but they're fundamentally different:

Aspect	HGAN	My Approach
What are nodes?	Image regions, text words (features)	Entire articles (documents)
What are edges?	Feature relationships within pairs	Article relationships across corpus
Graph scope	Single image-text pair	Entire 202K article corpus
Processing	Attention mechanisms (not GCN)	GNN (GraphSAGE/GAT)
Goal	Align features within pairs	Find related articles across database
Scale	Small (dozens of nodes per pair)	Large (202K nodes total)

Table 4: Key Differences Between HGAN and My Approach

5.4 My Complete Pipeline (Step-by-Step)

STAGE 1: Offline Graph Construction (Done Once Before Any Queries)

1. **Extract Article Features:**
   - Take all 202,803 articles
   - For each article, combine title + content into one text
   - Use pretrained Qwen3-Embedding to encode: article → 1024-dimensional vector
   - Store all vectors (total: 800MB storage)
2. **Build Temporal Edges:**
   - For each article, find all articles published within 14 days
   - Compute weight: $w = e^{-0.1 \times \text{days\_apart}}$ (closer = higher weight)
   - Add edge if weight greater than 0.1
3. **Build Entity Edges:**
   - Use spaCy to extract named entities from each article
   - For each pair of articles, compute: overlap = $\frac{\text{shared entities}}{\text{total entities}}$
   - Add edge if overlap greater than 0.3
4. **Build Semantic Edges:**
   - Compute cosine similarity between article vectors
   - Add edge if similarity greater than 0.7 (only very similar articles)
5. **Save Graph:**

- Store as PyTorch Geometric Data object
- Sparse format (average 30-50 edges per node)
- Total storage: approximately 2GB

STAGE 2: Train GNN (On OpenEvents Training Set)

1. **Architecture:**
   - Use 2-layer GraphSAGE or GAT from PyTorch Geometric
   - Input: Pretrained article embeddings (frozen - not updated during training)
   - GNN learns to propagate relevance through graph structure
   - Output: Graph-enhanced embeddings
2. **Training Process:**
   - Use training captions with ground-truth article labels
   - Loss function: Ranking loss (push relevant articles higher in ranking)
   - Train only on OpenEvents training split (fully compliant)
   - Validation on OpenEvents validation split

STAGE 3: Online Retrieval (When Query Comes In)

1. **Initial Dense Retrieval:**
   - Embed query caption with Qwen3-Embedding
   - Find top-100 articles by cosine similarity (same as EVENT-Retriever)
   - This gives us seed nodes
2. **Graph Expansion with GNN:**
   - Take top-100 articles as seed nodes in the graph
   - Use GNN to explore 1-2 hop neighbors
   - Message passing aggregates information from connected articles
   - This expands candidates from 100 to approximately 300 articles
   - **Key benefit:** We now include contextually related articles that had lower direct similarity
3. **Reranking:**
   - Apply Qwen3-Reranker on expanded set of approximately 300 articles
   - Select top-10 articles
4. **Image Collection and Scoring:**
   - Collect images from top-10 articles (at least 10 images from at least 3 articles)
   - Score with gme-Qwen2-VL
   - Apply rank-aware selection (prioritize images from higher-ranked articles)
   - Return top-10 images

5.5 Why This Should Work Better

Going back to our earlier example:

Traditional approach: Only retrieves Article A (high similarity)

My graph approach:

1. Retrieves Article A (high similarity) - same as before
2. Graph has edges: A ↔ B, A ↔ C, A ↔ D
3. GNN propagates relevance: A's high score flows to B, C, D
4. Now B, C, D are also in candidate set
5. We consider all their images, not just A's images
6. **Better chance of finding the best image**

6 Why This is Novel and Important

6.1 What Makes It Different

Aspect	Current SOTA	My Approach
How articles retrieved	Independent pairwise similarity	Graph-based with relationships
Article relationships	Not modeled	Explicitly modeled (temporal, entity, semantic)
Type of reasoning	Pairwise (query vs article)	Multi-hop through graph
Candidate expansion	Fixed top-K	Graph neighborhood exploration
Scale	N/A	First at 200K+ article scale
Graph usage	None or feature-level (HGAN)	Document-level across corpus

Table 5: Novelty Comparison with State-of-the-Art

6.2 My Specific Contributions

1. **First application of document-level Graph Neural Networks for event-based image retrieval at this scale (200K+ articles)**
   - HGAN used graphs but only at feature level within pairs
   - GNN paper tested on small datasets (few thousand samples)
   - Nobody has built article relationship graphs at this scale

2. **Explicit modeling of temporal, entity, and semantic article relationships**
    - Current methods rely entirely on learned embeddings
    - I explicitly encode known relationships (publication dates, shared entities)
    - This makes reasoning interpretable
3. **Graph-based expansion strategy that finds contextually related articles**
    - Current methods stop at top-K similarity
    - I use GNN message passing to explore the event context
    - Can find relevant articles with lower direct similarity
4. **Modular design that integrates with existing SOTA components**
    - Works with Qwen encoders, LLM rerankers, VLM scorers
    - Can be added to existing pipelines
    - Not a completely new system, just an enhancement

7 Challenge Compliance: Following the Rules

7.1 Closed Track Requirements

The closed track has strict rules to ensure fair comparison:

1. **Only use provided OpenEvents dataset** - No external training data allowed
2. **Only use publicly available pretrained models** - Models must be public before challenge start
3. **No external knowledge sources** - Can't use Wikipedia, knowledge graphs, etc.
4. **Must be reproducible** - Other people should be able to run my code

7.2 How My Approach Complies

Component	Compliance Details
Text Encoders	Qwen3-Embedding, BERT, Nomic - all public on HuggingFace
Image Encoders	CLIP, OpenCLIP - all public
Vision-Language Model	gme-Qwen2-VL - public
LLM Reranker	Qwen3-Reranker - public
Graph Construction	Built from OpenEvents data only (no external data)
Entity Extraction	spaCy (open-source tool)
GNN Training	Trained ONLY on OpenEvents training split
GNN Library	PyTorch Geometric (open-source)

Table 6: Detailed Compliance Check

Key clarification:

I'm using pretrained encoders to extract features (compliant), then building a graph structure from the provided dataset (compliant), then training a GNN only on OpenEvents training data (compliant). Everything follows the rules.

8 Implementation Plan: Making It Work on Kaggle

8.1 Kaggle Environment Specifications

Resource	Available
GPU Options	Tesla P100 (16GB) or T4 x2 (16GB each)
CPU	4 cores
RAM	29GB
Storage	Persistent datasets + temporary storage
Session Time Limit	12 hours per session

Table 7: Kaggle Hardware Resources

8.2 Feasibility Analysis

Can this actually run on Kaggle? Let me check:

Task	Memory Need	Feasible?
Store 202K article embeddings	800MB	Yes (29GB available)
Build sparse graph (avg 40 edges/node)	2GB	Yes
Train 2-layer GNN with batching	4-6GB	Yes
Inference (graph already built)	3GB	Yes
Full pipeline (encoding + retrieval)	8-10GB	Yes

Table 8: Memory Feasibility Check

Conclusion: This is definitely feasible on Kaggle's hardware.

9 Expected Results

9.1 Performance Targets

Method	mAP	R@1	R@5	R@10	Overall
EVENT-Retriever (Rank 1)	0.563	0.469	0.690	0.744	0.577
Hybrid (Rank 2)	0.558	0.456	0.698	0.762	0.572
Hierarchical (Rank 3)	0.549	0.449	0.695	0.738	0.564
My Target	**0.555+**	**0.460+**	**0.695+**	**0.750+**	**0.570+**

Table 9: Performance Targets (Aiming for Top 3)

9.2 Why I Expect Improvement

1. **Better Recall:**
   - Graph expansion should retrieve more relevant articles
   - Current methods miss contextually related articles with lower similarity
   - I should capture more of the event narrative
2. **Better Precision:**
   - GNN aggregates neighborhood information for better ranking
   - Articles connected to highly relevant articles get boosted
   - More informed decisions about which articles to keep
3. **Better Event Coverage:**
   - Temporal and entity edges naturally capture event progressions
   - Can follow the story across multiple articles
   - More likely to find the "best" image showing the key moment

9.3 Planned Ablation Studies

To prove the graph actually helps, I'll test different configurations:

Configuration	What It Tests
Dense retrieval only (no graph)	Baseline - current approach
Dense + graph expansion (no GNN learning)	Whether graph structure alone helps
Dense + GNN (only semantic edges)	Contribution of semantic edges
Dense + GNN (only temporal edges)	Contribution of temporal signal
Dense + GNN (only entity edges)	Contribution of entity signal
Dense + GNN (all edges)	Full proposed approach

Table 10: Planned Ablation Experiments

This will show exactly which components contribute to performance.

10 Conclusion

10.1 Summary

This project addresses a critical limitation in current event-based image retrieval systems: **they treat articles as independent entities and miss contextually related content.**

My solution:

- Build a graph connecting articles based on temporal proximity, entity overlap, and semantic similarity
- Use Graph Neural Networks to explore these connections during retrieval
- Find contextually related articles that pure similarity search overlooks
- Integrate seamlessly with existing SOTA components (Qwen, RRF, etc.)

10.2 Key Strengths

1. **Novel:** First corpus-wide GNN application for event-based retrieval at 200K+ scale
2. **Compliant:** Uses only pretrained public models and provided dataset (follows all rules)
3. **Feasible:** Can run on Kaggle GPU environment with available resources
4. **Modular:** Integrates with existing SOTA components, not a complete replacement
5. **Interpretable:** Graph edges provide explicit reasoning paths (can see why articles are connected)

10.3 The Core Insight

Events are not isolated - they unfold across multiple articles published over time. By capturing these relationships through graph structure, we can retrieve images from contextually relevant articles that pure embedding similarity overlooks.

This is especially important for news events where:

- Initial reports provide breaking news
- Follow-up articles provide analysis and reactions
- Later articles show consequences and developments
- All are part of the same story, but have different textual similarities to a given caption

Graph structure lets us explore the full event narrative, not just the articles with highest word overlap.

References

[1] H. Nguyen, P. Nguyen, T. Tran, M. Nguyen, T. V. Nguyen, M. Tran, and T. Le, *OpenEvents V1: Large-Scale Benchmark Dataset for Multimodal Event Grounding*, ACM Multimedia 2025, EVENTA Grand Challenge.

[2] Anonymous Authors, *EVENT-Retriever: Event-Aware Multimodal Image Retrieval for Realistic Captions*, EVENTA 2025 Challenge Track 2 (Rank 1, Overall Score: 0.5766).

[3] Anonymous Authors, *Hierarchical Article-to-Image: Leveraging Multi-Granularity Text Representations for Article Ranking and Text-Visual Similarity for Image Retrieval*, EVENTA 2025 Challenge Track 2 (Rank 3, Overall Score: 0.563).

[4] Anonymous Authors, *A Hybrid Dense-Sparse Multi-Stage Re-ranking Framework for Event-Based Image Retrieval*, EVENTA 2025 Challenge Track 2 (Rank 2, Overall Score: 0.572).

[5] J. Guo, M. Wang, Y. Zhou, B. Song, Y. Chi, W. Fan, and J. Chang, *HGAN: Hierarchical Graph Alignment Network for Image-Text Retrieval*, Journal of LaTeX Class Files, Vol. 14, No. 8, August 2021.

[6] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, *ImageBind: One Embedding Space To Bind Them All*, CVPR 2023.

[7] Y. Yuan and H. Xue, *Multimodal Information Integration and Retrieval Framework Based on Graph Neural Networks*, BDICN 2025: 4th International Conference on Big Data, Information and Computer Network, January 10-12, 2025, Guangzhou, China.