ATOM: AdapTive and OptiMized dynamic temporal knowledge graph construction using LLMs

Yassir LAIRGI[1,2], **Ludovic MONCLA**[1], **Khalid BENABDESLEM**[1],
Rémy CAZABET[1], and **Pierre CLÉAU**[2]

[1]INSA Lyon, CNRS, UCBL, LIRIS, UMR5205, 69621 Villeurbanne, France
[2]GAUC, Lyon, France
{ludovic.moncla, khalid.benabdeslem, remy.cazabet}@liris.cnrs.fr
{yassir.lairgi, pierre.cleau}@auvalie.com

Abstract

In today's rapidly expanding data landscape, knowledge extraction from unstructured text is vital for real-time analytics, temporal inference, and dynamic memory frameworks. However, traditional static knowledge graph (KG) construction often overlooks the dynamic and time-sensitive nature of real-world data, limiting adaptability to continuous changes. Moreover, recent zero- or few-shot approaches that avoid domain-specific fine-tuning or reliance on prebuilt ontologies often suffer from instability across multiple runs, as well as incomplete coverage of key facts. To address these challenges, we introduce ATOM[1] (AdapTive and OptiMized), a few-shot and scalable approach that builds and continuously updates Temporal Knowledge Graphs (TKGs) from unstructured texts. ATOM splits input documents into minimal, self-contained "atomic" facts, improving extraction exhaustivity and stability. Then, it constructs atomic TKGs from these facts, employing a dual-time modeling that distinguishes between when information is observed and when it is valid. The resulting atomic TKGs are subsequently merged in parallel. Empirical evaluations demonstrate that ATOM achieves $\sim 18\%$ higher exhaustivity, $\sim 33\%$ better stability, and over 90% latency reduction compared to baseline methods, demonstrating a strong scalability potential for dynamic TKG construction.

1 Introduction

Unstructured data is expanding at an unprecedented rate (Dresp-Langley et al., 2019), and given that the majority of big data is inherently unstructured (Trugenberger, 2015), there is an urgent need for robust information extraction and data modeling techniques to unlock its potential and derive insights across a broad spectrum of applications (Cetera et al., 2022). A prominent model for converting this unstructured data into structured, actionable knowledge is the Knowledge Graph (KG) (Zhong et al., 2023).

KG construction involves identifying entities, relationships, and attributes from diverse data sources to create structured knowledge representations. Traditionally, many approaches have focused on static KGs, which provide snapshots of knowledge without incorporating temporal dynamics. However, as real-world phenomena are inherently dynamic, static KGs, rarely or never updated, struggle to remain relevant and accurate (Jiang et al., 2023). In contrast, Temporal Knowledge Graphs (TKGs) integrate time dimensions by associating timestamps or time intervals with facts (e.g., (Einstein, was awarded, the Nobel Prize, in 1921)), making them particularly well-suited for analyzing changes, trends, and enabling temporal reasoning.

GraphRAG (Edge et al., 2024) and agent-based architectures (Xi et al., 2025) have demonstrated the potential of TKGs in retrieving and modeling dynamic information (Wu et al., 2024). Additionally, TKGs have been effectively used to model the memory of agents within agentic systems (Anokhin et al., 2025), highlighting their role in capturing the evolving nature of knowledge for adaptive and responsive systems.

Despite this potential, deploying Dynamic TKGs in production imposes latency constraints. As new information arrives continuously, systems must update the graph in real-time. In such contexts, sequential approaches often become prohibitive bottlenecks, creating a critical limitation for scalable, real-time analytics (Bian, 2025).

Traditional methods for KG construction, often reliant on entity recognition and relation extraction, face several limitations. They typically depend on predefined ontologies and supervised learning techniques that require extensive human annotation (Al-Moslmi et al., 2020). Recent advances in

---

[1]The code, prompts, and dataset are available at `https://github.com/AuvaLab/itext2kg`. ATOM is available as an open-source Python library.

Large Language Models (LLMs) (Jin et al., 2024) and zero- or few-shot techniques (Zhang et al., 2024; Carta et al., 2023; Hu et al., 2024) have paved the way for more flexible KG construction approaches that reduce dependency on extensive training datasets.

Despite these advances, current zero- or few-shot methods for KG construction often suffer from several limitations. They can be non-exhaustive, omitting key relationships, and prone to instability, where multiple construction runs on the same text yield different results. Moreover, many of these approaches overlook the temporal dimension of the input data and struggle to adapt to real-life scenarios with dynamic, evolving data, leading to false positives and a lack of scalability (Cai et al., 2024).

In this paper, we propose ATOM (AdapTive and OptiMized), a few-shot and scalable dynamic TKG construction approach from unstructured text, ensuring stability and exhaustivity. ATOM introduces a strategy that decomposes unstructured text into atomic facts. Rather than processing these atomic facts sequentially, ATOM proposes an architecture with parallel 5-tuple extraction, followed by a parallel atomic merging mechanism. In the remainder of the paper, we present related work in Section 2, our proposed approach, ATOM, in Section 3, experimental evaluation in Section 4, a conclusion in Section 5, and limitations in Section 6.

2 Related work

Current zero- and few-shot approaches to KG construction, such as AttacKG+ (Zhang et al., 2024), iterative LLM prompting pipelines (Carta et al., 2023), LLM-Tikg (Hu et al., 2024), LLM Builder [2], and LLM Graph Transformer[3] aim to build KGs without requiring task-specific training. However, these methods suffer from inconsistencies such as unresolved entities and relations.

That is why iText2KG (Lairgi et al., 2024) introduces an incremental, zero-shot architecture that constructs KGs iteratively by comparing newly extracted entities and relations with existing ones using embeddings and cosine similarity, achieving performance gains over some state-of-the-art LLM-based methods. However, it produces non-exhaustive and non-stable KGs due to the stochastic

nature of LLMs (Atil et al., 2024). Moreover, it fails to incorporate the temporal dimension, and scalability remains a significant challenge when applying it to real-world scenarios due to its incremental nature.

Graphiti (Rasmussen et al., 2025) proposed a dynamic TKG construction approach for agents' memory with an exclusively LLM-based entity/relation and temporal resolution framework. A key limitation is that it relies solely on prompting the LLM across all its modules, making it heavily dependent on LLM calls. The system prompts the LLM with all previous entities for entity resolution, which becomes impractical as the graph scales to millions of nodes. Similarly, time conflicts are resolved exclusively through LLM calls, resulting in high computational costs and scalability challenges when applied to large-scale datasets. Moreover, they do not handle the exhaustivity and stability of the constructed TKGs.

AriGraph (Anokhin et al., 2025) integrated semantic and episodic memories to support reasoning, planning, and decision-making in LLM agents. However, their entity resolution method leads to semantic drift in the temporal KG, where, for example, a reference to "Apple" might ambiguously denote either the company or the fruit. Furthermore, scalability becomes problematic as the volume of unstructured data increases.

Despite these advances, current zero- and few-shot TKG construction methods face three key limitations: (1) they struggle to maintain exhaustive fact coverage when processing longer texts, (2) they often produce unstable TKGs across multiple runs, and (3) they lack scalable architectures for dynamic temporal updates. To address these challenges, we propose ATOM, a framework that combines atomic fact decomposition for exhaustive and stable extraction and parallel merging for scalability.

3 Proposed approach: ATOM

In this section, we first present some notations and definitions used throughout the paper and then introduce the formulation of our proposed framework.

3.1 Problem statement

ATOM incorporates dual-time modeling, differentiating between when facts are observed and the temporal information conveyed by the facts themselves, which is characterized by a validity period.
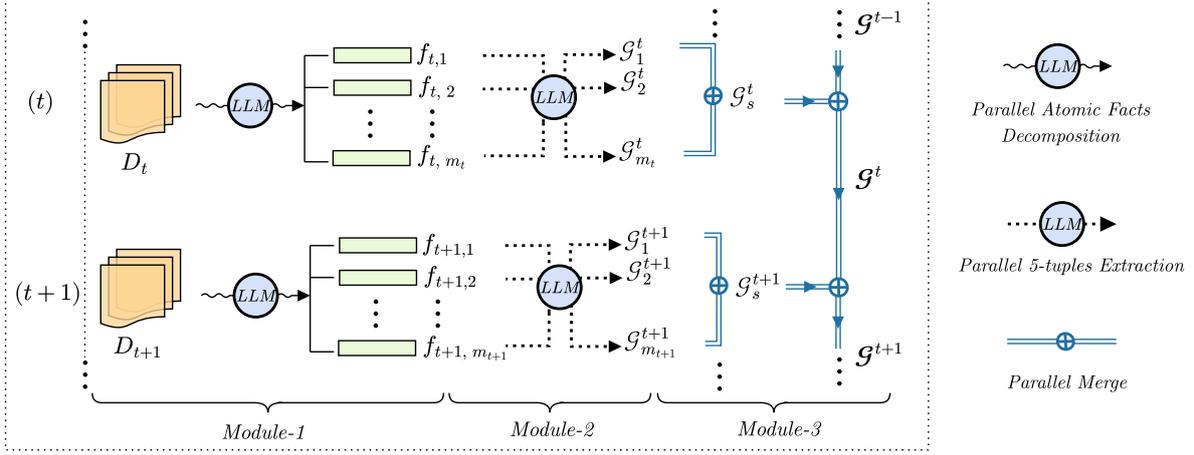
Figure 1: ATOM's architecture, running in parallel, ensuring scalability, speed, and continuous updates. Unstructured texts observed at time t are denoted by D_t, the i-th temporal atomic fact observed at time t is denoted by $f_{t,i}$, the i-th atomic TKG observed at time t is denoted by $\mathcal{G}_i^t$, the TKG snapshot observed at time t is denoted by $\mathcal{G}_s^t$, and the updated TKG at time t is denoted by $\mathcal{G}^t$.

This approach better reflects real-world data (Rasmussen et al., 2025; Chekol and Stuckenschmidt, 2018; Meijer, 2022). This separation enhances TKG dynamism and proper inference of relative times by providing the observation time as context to the LLM (eg, 'month ago').

Definition 1 (Dynamic-Temporal KG with Dual–Time Modeling). *Let $\mathcal{T}_{obs}$ be an ordered set of observation timestamps at which the KG is updated, and let $\mathcal{T}_{start}$, $\mathcal{T}_{end}$ be sets of timestamps used to label inherent validity period of facts defined by their start and end times, respectively. For each observation time $t \in \mathcal{T}_{obs}$, a TKG snapshot is defined as:*

$$\mathcal{G}_s^t = \left(\mathcal{E}^t, \; \mathcal{R}^t, \; \mathcal{T}_{start}^t, \; \mathcal{T}_{end}^t, \; \mathcal{F}^t \right) \quad (1)$$

where:

- $\mathcal{E}^t$ *is the set of entities known at* observation time t,

- $\mathcal{R}^t$ *is the set of relations known at* observation time t,

- $\mathcal{T}_{start}^t \subseteq \mathcal{T}_{start}$ *is the set of* validity start times *referenced by facts in this snapshot,*

- $\mathcal{T}_{end}^t \subseteq \mathcal{T}_{end}$ *is the set of* validity end times *referenced by facts in this snapshot,*

- $\mathcal{F}^t \subseteq \mathcal{E}^t \times \mathcal{R}^t \times \mathcal{E}^t \times \mathcal{T}_{start}^t \times \mathcal{T}_{end}^t$ *is the set of temporal facts (5-tuples) observed in the snapshot at* observation time t.

A fact in this snapshot is a 5-tuple (quintuple) $(e_s, r_p, e_o, t_{start}, t_{end})$ indicating the relation $r_p \in \mathcal{R}^t$ holds between the subject entity $e_s \in \mathcal{E}^t$ and the object entity $e_o \in \mathcal{E}^t$. Technically, t_{start} and t_{end} are chosen to be lists to aggregate start and end validity timestamps to track the history of the same fact. Validity start and end timestamps can be unknown, in which case their respective lists are empty (denoted as $[.]$).

A Dynamic Temporal Knowledge Graph (DTKG), updated at t, is defined as the parallel pairwise merge of these TKG snapshots via the merge operator $\oplus$ (as described later in Section 3.2.3).

$$\mathcal{G}^t = \bigoplus_{t' \in \mathcal{T}_{obs} = \{\dots, t-1, t\}} \mathcal{G}_s^{t'} = \mathcal{G}^{t-1} \oplus \mathcal{G}_s^t \quad (2)$$

Definition 2 (Temporal Atomic Fact with Dual–Time Modeling). *Let $\mathcal{T}_{obs}$ be a set of observation timestamps at which new data is ingested, and let $\mathcal{T}_{start}$ and $\mathcal{T}_{end}$ be the sets of validity periods mentioned within the data. For each observation time $t \in \mathcal{T}_{obs}$, let D_t be an unstructured text that becomes available at t. A temporal atomic fact $f_{t,i}$ is a short, self-contained snippet derived from D_t that conveys exactly one piece of information, extracted by the LLM. Depending on the content of the snippet, an atomic fact may or may not explicitly contain a validity period. Formally,*

$$\mathsf{ExtractAFacts}_{\mathsf{LLM}}(D_t) = \{f_{t,1}, \dots, f_{t,m_t}\} \quad (3)$$

An example is provided in Section B in the Appendices. In what follows, the term atomic fact is used

for conciseness.

Definition 3 (Atomic Temporal KG). *Given an atomic fact $f_{t,i}$ observed at time t, its atomic temporal KG $\mathcal{G}_i^t$ is the set of 5-tuples extracted by the LLM:*

$$\mathcal{G}_i^t = \mathsf{ExtractQuintuples}_{\mathsf{LLM}}(f_{t,i}) \qquad (4)$$
$$\subseteq \mathcal{P}\big(\mathcal{E}^t \times \mathcal{R}^t \times \mathcal{E}^t \times \mathcal{T}_{start}^t \times \mathcal{T}_{end}^t\big)$$

Concretely, $\mathcal{G}_i^t$ is the set of 5-tuples $(e_s, r_p, e_o, t_{start}, t_{end})$ derived from a single atomic fact $f_{t,i}$ at observation time t.

Given the definitions 1 and 3, the DTKG, updated at t:

$$\mathcal{G}^t = \bigoplus_{t' \in \mathcal{T}_{obs}} \mathcal{G}_s^{t'} = \bigoplus_{t' \in \mathcal{T}_{obs}} \big(\bigoplus_{i \in [\![1, m_{t'}]\!]} \mathcal{G}_i^{t'} \big) \quad (5)$$

Figure 2 illustrates a detailed example of ATOM's pipeline.

3.2 ATOM's Framework

Given a continuous stream of unstructured texts, our goal is to construct and maintain a consistent and dynamic TKG, ensuring for each $t \in \mathcal{T}_{obs}$:

(C1) Exhaustivity: the constructed TKG snapshot $\mathcal{G}_s^t$ ideally captures every 5-tuple that is semantically present in D_t.

(C2) Stability across multiple runs: when the identical 5-tuple extraction prompt is executed repeatedly on the same input text using the same LLM, the resulting TKG snapshots should be nearly identical.

In the following, we detail the different modules of ATOM's architecture (Figure 1).

3.2.1 Module-1: Atomic fact decomposition

ATOM does not construct TKGs directly from raw input documents but first decomposes them into atomic facts (Figure 1). This decomposition addresses a fundamental limitation of LLMs: the "forgetting effect" where models prioritize salient information in longer contexts while omitting key relationships, leading to incomplete knowledge extraction (Liu et al., 2024). Following (Hosseini et al., 2024; Chen et al., 2024; Raina and Gales, 2024), ATOM uses LLM-based prompting for decomposition with an optimal chunk size to maintain high exhaustivity (determined experimentally

in Section 4.3). However, while prior work focused on information retrieval applications, ATOM applies atomic decomposition specifically for TKG construction. This strategy addresses both conditions: it enhances exhaustivity (C1) by preventing information loss that occurs when LLMs process complex, multi-fact paragraphs, and enhances stability (C2) by providing clear, unambiguous contexts that reduce output variance across multiple runs. Each atomic fact is related to an observation time, and it is necessary to encapsulate the relative validity period presented in the context. The primary computational challenge of this approach is scale: a single document can yield hundreds or thousands of atomic facts. Sequential processing of each fact for 5-tuple extraction, followed by entity/relation and temporal resolution, becomes time-consuming. To address this challenge, ATOM employs a parallel architecture for both extraction and merging phases, as detailed in the subsequent modules.

3.2.2 Module-2: Atomic TKGs construction

5-tuples are extracted from each atomic fact in parallel using an LLM, producing atomic TKGs $\mathcal{G}_i^t$ while embedding nodes and relations following (Lairgi et al., 2024). To facilitate temporal resolution in Module-3, ATOM preprocesses 5-tuples during their extraction. It prevents separate quintuples describing the same temporal fact from co-existing in the same TKG such as (John_Doe, is_ceo, X, [01-01-2025], [.]) and (John_Doe, is_no_longer_ceo, X, [01-01-2026], [.]), which should be resolved into (John_Doe, is_ceo, X, [01-01-2025], [01-01-2026]). During the extraction, few-shot examples are provided as context to the LLM to transform end validity facts into affirmative counterparts while modifying only the t_{end} time. For instance, the statement John Doe is no longer the CEO of X on 01-01-2026 is converted into the 5-tuple (John_Doe, is_ceo, X, [.], [01-01-2026]), ensuring direct matching with the corresponding validity start time 5-tuple during the merge. For relative temporal expressions (e.g., 'a month ago'), the observation time is provided as context, enabling the LLM to infer the validity period.

3.2.3 Module-3: Parallel atomic merge of TKGs and DTKG update

ATOM then employs the binary merge algorithm (Algorithm A.1 in the Appendices) to merge pairs of atomic TKGs. The algorithm proceeds in three phases: first, entity resolution searches for exact
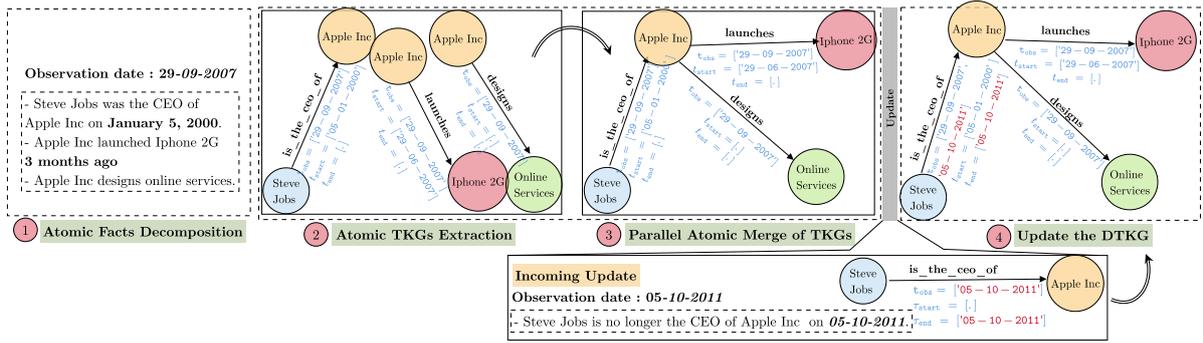
Figure 2: Example overview of ATOM's pipeline. It begins with atomic fact decomposition, followed by the extraction of atomic TKGs from these facts, which are then merged in parallel. When an incoming update arrives, ATOM handles the temporal resolution by transforming the end action into the affirmative part while modifying only the t_{end}, then merges the resulting atomic TKG with the existing DTKG.

matches between $\mathcal{G}_i^t$ and $\mathcal{G}_{i+1}^t$ based on name and label. When no exact match exists, cosine similarity is computed, merging entities if similarity exceeds θ_E. Second, relation resolution merges relation names regardless of endpoints and timestamps (e.g., owns $\longleftrightarrow$ possesses $\longleftrightarrow$ has) using threshold θ_R. Third, temporal resolution merges observation and validity time sets for relations with similar (e_s, r_p, e_o), detecting and aligning end-action facts with their corresponding beginning facts. Unlike Graphiti, ATOM avoids LLM calls during merging, improving scalability and preventing context overflow as the graph expands. The preprocessing of end-actions during extraction enables this LLM-independent merging approach. Subsequently, the binary merge function is extended to handle the entire set of atomic TKGs through iterative pairwise merging in parallel until a single consolidated TKG is obtained (Algorithm A.2 in the Appendices). Atomic TKGs are organized into pairs, with each pair merged in parallel. If the number of TKGs is odd, the remaining TKG carries forward to the next iteration. This process continues iteratively, reducing the number of TKGs at each step, until convergence to a single merged TKG. This parallel strategy scales with the number of available threads and addresses the computational challenge from Module-1, enabling ATOM to maintain low latency while preserving the exhaustivity and stability benefits of atomic decomposition. After the merge of all atomic TKGs, the snapshot $\mathcal{G}_s^t$ is obtained, and it is merged with the previous DTKG $\mathcal{G}^{t-1}$ using Algorithm A.1 to yield the DTKG updated at t, $\mathcal{G}^t$.

4 Experiments

Our evaluation addresses the following research questions:

RQ1: How does exhaustivity deteriorate as the LLM context increases, and what degree of information loss could occur?

RQ2: How does ATOM's atomic fact decomposition enhance stability, exhaustivity, and improve the quality of the 5-tuples?

RQ3: How does ATOM scale with the number of atomic facts provided as input, and what is its time complexity compared to baseline methods?

RQ4: How does ATOM perform on DTKG construction consistency compared to baseline methods?

4.1 Metrics

To assess (C1), the performance is evaluated at two levels: factual and factual-temporal. Given a gold-standard KG, factual true positives (TP_f) denote correctly extracted triplets, false negatives (FN_f) refer to triplets that exist in the source but are missing from the extracted KG (omission), and false positives (FP_f) represent unsupported triplets (hallucination). These definitions are extended to the factual-temporal level for 5-tuples $(e_s, r_p, e_o, t_{\text{start}}, t_{\text{end}})$. A 5-tuple is considered valid only if it exhibits correctness at both the factual and temporal levels. Hence, the factual-temporal true positives $\text{TP}_{f,t}$ are 5-tuples whose t_{start} and t_{end} are both correct. Consequently, $\text{TP}_{f,t} \subseteq \text{TP}_f$.

Exhaustivity (RQ1 & RQ2). The exhaustivity or recall is reported at two levels of strictness. The exhaustivity of the factual component:

$$R_f = \frac{|\text{TP}_f|}{|\text{TP}_f| + |\text{FN}_f|} \qquad (6)$$

The exhaustivity of the factual-temporal component $R_{f,t}$ is computed analogously with $|\text{TP}_{f,t}|$ in the numerator. It measures the proportion of the total gold standard 5-tuples that are both factually and temporally correct.

5-tuples quality (RQ2). It is assessed through Recall, Precision, and F_1-score. The precision of the factual component:

$$P_f = \frac{|\text{TP}_f|}{|\text{TP}_f| + |\text{FP}_f|} \qquad (7)$$

The precision of the factual-temporal component $P_{f,t}$ is computed similarly to P_f with $|\text{TP}_{f,t}|$ in the numerator. It measures the proportion of all extracted 5-tuples that are both factually and temporally correct. The $F_{1,f}$ and $F_{1,f,t}$ scores are computed as the harmonic means of their respective precision and recall metrics.

Stability (RQ2). It is measured using the Jaccard similarity between the 5-tuple sets obtained across independent runs. Let $\boldsymbol{s}^{(1)}$ denote the set of 5-tuples obtained during the baseline run (RUN 1) and let $\boldsymbol{s}^{(r)}$ denote the set of 5-tuples obtained at repetition r. Formally, the stability score S_r is computed as:

$$S_r = J\!\left(\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(r)}\right) = \frac{\left|\boldsymbol{s}^{(1)} \cap \boldsymbol{s}^{(r)}\right|}{\left|\boldsymbol{s}^{(1)} \cup \boldsymbol{s}^{(r)}\right|} \qquad (8)$$

This score is computed for repetitions $r = 2, 3$.

Time complexity (RQ3). Time complexity is evaluated by progressively increasing the number of atomic facts provided as input and measuring the total wall-clock latency required to construct the complete DTKG.

DTKG consistency (RQ4). For entity/relation resolution, the false discovery rate (1-precision) is defined in (Lairgi et al., 2024). Since they overlook recall and F_1-score, we extend the evaluation to include precision (P), recall (R), and F_1-score for both entity resolution (ER) and relation resolution (RR), denoted as Metric_{ER} and Metric_{RR}, respectively. For temporal resolution, a qualitative comparison is provided.

4.2 Datasets and baseline methods

Identifying a suitable and publicly available dataset for evaluating temporal extraction remains a significant challenge due to the task's specific structural requirements. While DocRed (Yao et al., 2019) is a standard benchmark for relation extraction, it is largely unsuitable for temporal analysis owing to documented labeling inconsistencies (Tan et al., 2022). Although TempDocRed (Zhu et al., 2025) introduces temporal layers, its scope remains limited, focusing exclusively on named entities and event start dates. Furthermore, the CS-GS and Music-GS datasets (Kabal et al., 2024), lack the necessary temporal dimensions. Consequently, this study adopts the NYT News dynamic and temporal dataset (Singh, 2023), which provides extensive temporal coverage through two million lead paragraphs spanning from 2000 to the present. News articles provide diverse temporal dynamics, including events with specific start and end times, evolving situations, and relative temporal expressions that are representative of many real-world scenarios requiring TKG construction. From this dataset, the 2020-COVID-NYT subset comprising 1,076 articles that focus on COVID-19 dynamics during 2020 is extracted. This subset is enriched with human-verified atomic facts and 5-tuples. Publication dates are used as observation dates. Comprehensive details regarding the annotation guidelines, observation time modeling, and dataset statistics are provided in the Appendices (Sections E, D, and Table T.1). To the best of our knowledge, an approach for resolving duplicate entities and relations while maintaining KG consistency among the SOTA methods for zero- and few-shot KG construction is supported only by iText2KG and Graphiti. The temporal aspect is handled by Graphiti only. Hence, it is the primary comparator of ATOM. In all experiments, the temperature is set to 0 to prefer deterministic outputs and minimize stochasticity. `text-embedding-large-3`[4] is used for embeddings. $\theta_E = 0.8$ and $\theta_R = 0.7$ are estimated following (Lairgi et al., 2024) (details are in Section C in the Appendices).

4.3 Exhaustivity deterioration in longer contexts

Exhaustivity is evaluated by iteratively concatenating lead paragraphs (increas-

---

[4] https://platform.openai.com/docs/models/text-embedding-3-large

ing context size) and testing five SOTA LLMs `claude-sonnet-4-2025-01-31`[5], `gpt-4o-2024-11-20`[6], `mistral-large-2411`[7], `gpt-4.1-2025-04-14`[8], `o3-mini-2025-01-31`[9]. Figure 3 shows a clear "forgetting effect". A decreased factual and factual-temporal exhaustivity as token count increases across all models except `claude-Sonnet-4-2025-01-31`, which maintains the highest exhaustivity for atomic facts but degrades for 5-tuples. This indicates that LLMs prioritize salient facts in longer texts. Moreover, all models show higher exhaustivity for atomic fact decomposition than 5-tuple extraction. Atomic facts require surface-level parsing, while 5-tuples demand deeper semantic understanding of entities, relationship identification, and temporal extraction (t_{start}, t_{end}). This added complexity causes greater information loss. To mitigate information loss in the atomic fact decomposition, we empirically determine the optimal chunk size at < 400 tokens to keep the exhaustivity > 0.8. For subsequent evaluations, we use `claude-sonnet-4-2025-01-31` for atomic fact decomposition based on its superior performance. For 5-tuple extraction, both `gpt-4.1-2025-04-14` and `claude-sonnet-4-2025-01-31` perform comparably, hence `gpt-4.1-2025-04-14` is used due to its lower cost. Section 4.5 evaluates exhaustivity gains from using atomic facts as input.

4.4 TKG stability

Rerunning the same prompt multiple times leads to variations in output. To evaluate the stability score, 5-tuples are initially extracted from both atomic facts (denoted as (F) in Table 1) and lead paragraphs individually, establishing a baseline named Run 1. This extraction process is subsequently repeated twice more without altering any parameters. As shown in Table 1, the 5-tuple sets extracted from atomic facts exhibit high Jaccard similarity with the baseline across runs. In contrast, the extraction from lead paragraphs demonstrates significantly lower overlap, revealing high variability. This addresses (C2) and reflects the effect of atomic facts in maintaining a stable construction of TKGs with

[5]https://docs.claude.com/en/docs/about-claude/models/overview
[6]https://platform.openai.com/docs/models/gpt-4o-2024-11-20
[7]https://docs.mistral.ai/getting-started/models/models_overview/
[8]https://openai.com/index/gpt-4-1/
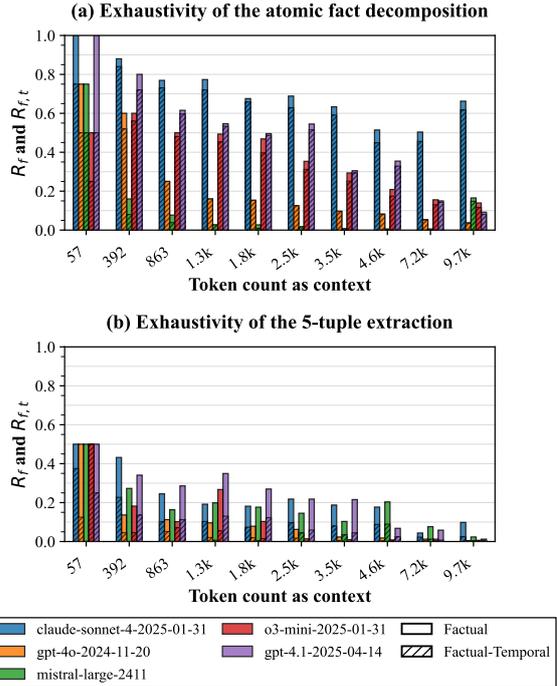[9]https://openai.com/index/openai-o3-mini/

Figure 3: Exhaustivity vs. token count as context for (a) the atomic fact decomposition (b) the 5-tuple extraction. The x-axis displays the context size (lead paragraphs) only; total input size includes additional system prompt tokens ($\sim$1.4k for (a) and $\sim$1.7k for (b)).

a gain of $\sim 33\%$.

Table 1: Stability S_r evaluated by rerunning the 5-tuple extraction process multiple times without any modifications, using `gpt-4.1-2025-04-14` with Run 1 as a baseline. The extraction is performed on (F) atomic facts and (L) lead paragraphs.

Dataset	Run 2	Run 3
2020-COVID-NYT (F)	**0.552 ± 0.124**	**0.534 ± 0.12**
2020-COVID-NYT (L)	0.212 ± 0.181	0.218 ± 0.195

4.5 The exhaustivity and quality of the 5-tuples

5-tuples are extracted from both atomic facts (denoted as (F) in Table 2) and lead paragraphs (L). As shown in Table 2, ATOM significantly outperforms the baseline in exhaustivity, with factual recall R_f increasing by $\sim 31\%$ and factual-temporal recall $R_{f,t}$ by $\sim 18\%$. These gains lead to a substantial improvement in overall extraction quality, resulting in increased factual $F_{1,f}$ and factual-temporal $F_{1,f,t}$ scores. Hence, atomic fact decomposition effectively mitigates information loss, addressing (C1). However, this improved exhaustivity presents a trade-off: a decrease in factual precision

P_f of $\sim 9\%$. This occurs because the baseline (L) acts conservatively, extracting only the most obvious facts, whereas ATOM's decomposition forces the LLM to explicitize context. This process can generate 'inferred' atomic facts that are semantically plausible but not strictly present in the gold standard, leading to false positives. Furthermore, a marginal decrease of $\sim 1\%$ in factual-temporal precision $P_{f,t}$ is observed. This is attributed to imperfections during the decomposition phase, where the LLM may fail to assign temporal information to certain atomic facts. Consequently, this error propagates to the 5-tuple extraction. This trade-off is discussed further in Section 6.

Table 2: 5-tuple quality metrics. The extraction is performed on (L) lead paragraphs and (F) atomic facts.

Metric	2020-COVID-NYT (L)	2020-COVID-NYT (F)
P_f	**0.667 ± 0.172**	0.572 ± 0.128
R_f	0.405 ± 0.150	**0.720 ± 0.143**
$F_{1,f}$-score	0.504	**0.638**
$P_{f,t}$	**0.292 ± 0.182**	0.281 ± 0.130
$R_{f,t}$	0.176 ± 0.123	**0.354 ± 0.165**
$F_{1,f,t}$-score	0.220	**0.313**

4.6 ATOM's time complexity

Given the demonstrated benefits of atomic fact decomposition in improving exhaustivity and stability, all subsequent experiments utilize atomic facts as input rather than lead paragraphs. All baseline methods are run using `gpt-4.1-2025-04-14`. ATOM employs 8 threads and a batch size of 40 atomic facts for 5-tuple extraction, which respects OpenAI rate limits. iText2KG and Graphiti separate entity and relation extraction, increasing latency. Graphiti's incremental entity/relation resolution, which relies on the LLM, limits parallel requests and significantly increases latency as the graph expands. Similarly, iText2KG is incremental, restricting parallel requests. Although iText2KG uses a distance metric for resolution, reducing LLM dependency, its separate extraction steps double the number of LLM calls and induce isolated entities that require further LLM iterations. Conversely, ATOM's architecture facilitates (1) parallel LLM calls, (2) parallel merge of atomic TKGs, (3) LLM-independent merging, and (4) temporal resolution. This design reduces latency by 93.8% compared to Graphiti and 95.3% compared to iText2KG (Figure 4). ATOM's Module-3 accounts for only 13% of its total latency, with the remainder attributed to API calls, which can be further minimized through

either increasing the batch size (by upgrading the API tier) or scaling hardware for local LLM deployment.
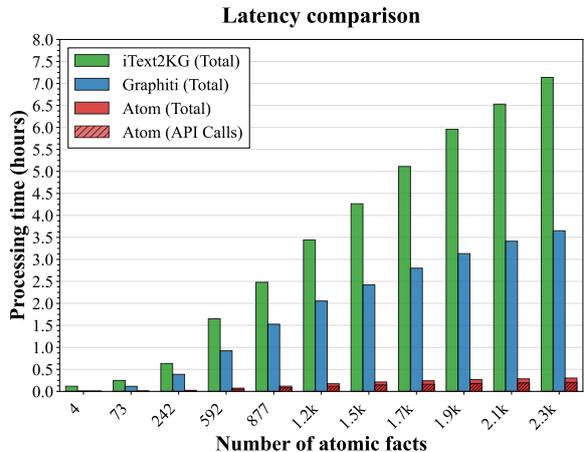

Figure 4: Latency comparison of the baseline methods as a function of the number of atomic facts as input.

4.7 Consistency of ATOM's DTKG construction

Table 3 shows that ATOM and iText2KG demonstrate comparable entity and relation resolution performance, as both employ a distance metric for merging. ATOM shows an improvement over Graphiti, whose incremental, LLM-based entity and relation resolution degrades with graph expansion (increasing context size), which is consistent with findings in Section 4.3. Beyond entity and relation resolution, temporal resolution reveals more significant differences between ATOM and Graphiti. The examples in Figures F.2 and F.3 in the Appendices illustrate atomic facts observed at different times that refer to the same information but with different validity periods. In these examples, Graphiti creates separate relations for 5-tuples with different validity periods (t_{start}, t_{end}), while ATOM detects these similar relations and extends their validity period history. This temporal resolution serves two functions: tracking events that naturally appear and disappear over time, and matching relations with only t_{start} or only t_{end} to complete their validity periods as additional information becomes available. Additionally, Graphiti incorporates validity periods only and does not allow for observation time modeling, treating observation time as t_{start}. This can induce errors. For example, if a news article observed on "January 23, 2020" states "The mysterious respiratory virus spread to at least 10 other countries," Graphiti would set $t_{start} = 23\text{-}01\text{-}2020$, while the statement does not specify a validity period and

the true validity could be weeks or days before its publication. In contrast, ATOM separately models observation time and validity periods, allowing it to recognize atomic facts without a validity period and avoid incorrect temporal assignments (Example in Figure F.1 in the Appendices).

Table 3: Performance on DTKG construction.

Metric	ATOM	Graphiti	iText2KG
P_{ER}	0.994	0.967	0.974
R_{ER}	0.993	0.952	0.980
$F_{1,ER}$-score	0.994	0.959	0.977
P_{RR}	1	0.917	0.991
R_{RR}	1	0.888	0.988
$F_{1,RR}$-score	1	0.902	0.989

5 Conclusion

In this paper, we presented ATOM, a few-shot and scalable approach for constructing and dynamically updating TKGs from unstructured texts. Experimental results indicate that ATOM's atomic fact decomposition effectively addresses the exhaustivity and stability challenges often observed in LLM-based TKG construction methods. Its parallel architecture accelerates TKG construction and enables scalability for larger unstructured texts. Potential directions for future work include quantitative evaluation of temporal resolution and fine-tuning an LLM specifically for refining atomic fact decomposition. In summary, ATOM enables fast and continuous updates of TKGs.

6 Limitations

ATOM has some limitations that warrant consideration. First, the atomic fact decomposition can introduce error propagation: the LLM may generate inferred facts not present in the source text, leading to increased hallucination, and may fail to properly assign temporal information to atomic facts, resulting in a decrease in the factual-temporal precision (Section 4.5). A potential improvement consists of fine-tuning an LLM model specifically for this task. Second, the distance-metric-based merging approach, while scalable and efficient, can occasionally merge semantically distinct named entities that exhibit high similarity (e.g., gpt-5:model and gpt-3.5:model). A supervised entity/relation resolution classifier trained on labeled entity pairs could replace the threshold-based approach. Third, ATOM entails higher costs than direct extraction (Table T.2 in the Appendices) due to the added decomposition step and the increased output tokens required to maintain high exhaustivity. This cost could be mitigated by using or fine-tuning open-weight models specifically for extracting 5-tuples from atomic facts.

7 Ethics statement

We utilized AI assistants exclusively for grammatical corrections and text polishing. All scientific claims, experimental designs, and original ideas are solely those of the authors.

References

Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L. Opdahl, and Csaba Veres. 2020. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881.

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. 2025. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 12–20. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, and 1 others. 2024. Non-determinism of" deterministic" llm settings. *arXiv preprint arXiv:2408.04667*.

Haonan Bian. 2025. Llm-empowered knowledge graph construction: A survey. *arXiv preprint arXiv:2510.20345*.

Li Cai, Xin Mao, Yuhao Zhou, Zhaoguang Long, Changxu Wu, and Man Lan. 2024. A survey on temporal knowledge graph: Representation learning and applications. *arXiv preprint arXiv:2403.04782*.

Salvatore Carta, Alessandro Giuliani, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023. Iterative zero-shot LLM prompting for knowledge graph construction. *arXiv preprint arXiv:2307.01128*.

Wiesław Cetera, Włodzimierz Gogołek, Aleksander Żołnierski, and Dariusz Jaruga. 2022. Potential for the use of large unstructured data resources by public innovation support institutions. *Journal of Big Data*, 9(1):46.

Melisachew Wudage Chekol and Heiner Stuckenschmidt. 2018. Towards probabilistic bitemporal knowledge graphs. In *Companion Proceedings of the The Web Conference 2018*, pages 1757–1762.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics.

Birgitta Dresp-Langley, Ole Kristian Ekseth, Jan Fesl, Seiichi Gohshi, Marc Kurz, and Hans-Werner Sehring. 2019. Occam's razor for big data? on detecting quality in large unstructured datasets. *Applied Sciences*, 9(15):3065.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Mohammad Javad Hosseini, Yang Gao, Tim Baumgärtner, Alex Fabrikant, and Reinald Kim Amplayo. 2024. Scalable and domain-general abstractive proposition segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8856–8872, Miami, Florida, USA. Association for Computational Linguistics.

Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. 2024. Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145:103999.

Xuhui Jiang, Chengjin Xu, Yinghan Shen, Xun Sun, Lumingyuan Tang, Saizhuo Wang, Zhongwu Chen, Yuanzhuo Wang, and Jian Guo. 2023. On the evolution of knowledge graphs: A survey and perspective. *arXiv preprint arXiv:2310.04835*.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8622–8642.

Othmane Kabal, Mounira Harzallah, Fabrice Guillet, and Ryutaro Ichise. 2024. Enhancing domain-independent knowledge graph construction through openie cleaning and llms validation. *Procedia Computer Science*, 246:2617–2626.

Yassir Lairgi, Ludovic Moncla, Rémy Cazabet, Khalid Benabdeslem, and Pierre Cléau. 2024. itext2kg: Incremental knowledge graphs construction using large language models. In *International Conference on Web Information Systems Engineering*, pages 214–229. Springer.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Lisa Meijer. 2022. Bi-vaks: Bi-temporal versioning approach for knowledge graphs. *Delft University of Technology*.

Vatsal Raina and Mark Gales. 2024. Question-based retrieval using atomic units for enterprise RAG. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 219–233, Miami, Florida, USA. Association for Computational Linguistics.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Aryan Singh. 2023. NYT Articles: 2.1M+ (2000-Present) Daily Updated. https://www.kaggle.com/datasets/aryansingh0909/nyt-articles-21m-2000-present. Accessed: 2025-06-01.

Qingyu Tan, Lu Xu, Lidong Bing, Hwee Tou Ng, and Sharifah Mahani Aljunied. 2022. Revisiting DocRED - addressing the false negative problem in relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8472–8487, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Carlo A Trugenberger. 2015. Scientific discovery by machine intelligence: A new avenue for drug research. *arXiv preprint arXiv:1506.07116*.

Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. Retrieval augmented generation for dynamic graph modeling. *arXiv preprint arXiv:2408.14523*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Yongheng Zhang, Tingwen Du, Yunshan Ma, Xiang Wang, Yi Xie, Guozheng Yang, Yuliang Lu, and Ee-Chien Chang. 2024. AttacKG+: Boosting attack knowledge graph construction with large language models. *arXiv preprint arXiv:2405.04753*.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62.

Jun Zhu, Yan Fu, Junlin Zhou, and Duanbing Chen. 2025. A temporal knowledge graph generation dataset supervised distantly by large language models. *Scientific Data*, 12(1):734.

A ATOM's algorithms

ATOM's framework is based on two main algorithms, presented below: Algorithm A.1 for merging pairs of TKGs and Algorithm A.2 for parallelizing the merge of lists of TKGs.

B Example of the atomic fact decomposition

Example: *(Observed in 01-01-2025) On June 18, 2024, Real Madrid won the Champions League final with a 2-1 victory. Following the triumph, fans of Real Madrid celebrated the Champions League victory across the city.*

- *Real Madrid won the Champions League final match on June 18, 2024.* (Observation $t_{obs} = [01 - 01 - 2025]$, $t_{\text{start}} = [18 - 06 - 2024]$, $t_{\text{end}} = [.]$)

- *The Champions League final match ended with a 2-1 victory for Real Madrid on June 18, 2024.* (Observation $t_{obs} = [01 - 01 - 2025]$, $t_{\text{start}} = [18 - 06 - 2024]$, $t_{\text{end}} = [.]$)

- *Fans of Real Madrid celebrated the Champions League final match victory across the city on June 18, 2024.* (Observation $t_{obs} = [01 - 01 - 2025]$, $t_{\text{start}} = [18 - 06 - 2024]$, $t_{\text{end}} = [.]$)

C Estimating the merging thresholds

The merging thresholds θ_R and θ_E were estimated by (Lairgi et al., 2024), based on the mean cosine similarity of 1,500 pairs of similar entities and relation names generated by `gpt-4-0613`[10]; however, because entity typology is not considered, a hybrid similarity measure is proposed, combining the entity name embedding and the entity label embedding as: $\lambda \cdot \text{embeddings}_{\text{name}} + \beta \cdot \text{embeddings}_{\text{label}}$. Using this measure, 1,200 pairs of similar entities incorporating typology were generated, and λ is optimized to maximize the resulting cosine similarity, after which it is determined that $\lambda = 0.8$, $\beta = 0.2$, and $\theta_E = 0.8$, while $\theta_R = 0.7$ is retained as previously estimated in (Lairgi et al., 2024).

---

[10] https://platform.openai.com/docs/models/gpt-4

D Observation time modeling

Modeling the observation time is essential both for capturing the dynamism of the TKG and for inferring relative times. For historical or retrospective unstructured text streams (e.g., past news articles or archive documents), the observation time should correspond to the original publication time rather than the time at which the document was processed and ingested into the DTKG. This distinction is essential for preserving the correct temporal ordering of events and enabling reliable inference of relative times.

Conversely, for prospective or continuously monitored sources, where data is ingested automatically as it becomes available, the observation time can be treated as the ingestion time. In such settings, ingestion reflects the earliest feasible moment at which the information could be known by the system.

The granularity of observation time is application-dependent and may be defined according to user requirements. For instance, COVID-19 news was simulated using daily observation snapshots, whereas social media streams may require a per-post snapshot.

E Annotation guidelines for 2020-COVID-NYT

The ground truth for the 2020-COVID-NYT dataset was constructed through a two-step semi-manual annotation process utilizing the Claude Sonnet 4 chat model with extended thinking[11]. We selected this model after observing its superior performance on a small subset of lead paragraphs compared to other available SOTA LLMs. Human annotators supervised the process, provided with the lead paragraph and an associated observation date (t_{obs}), and were instructed to follow the guidelines detailed below.

E.1 Phase 1: atomic fact decomposition

In this phase, the model decomposed complex paragraphs into atomic, self-contained, and temporally grounded statements. Annotators were tasked with verifying and correcting the exhaustivity of the decomposition, ensuring the absence of hallucinatory facts, and validating the correctness of validity times for temporal facts, strictly following these guidelines:

---

[11] https://claude.ai/

Algorithm A.1 Binary Merge of TKGs

1: **function** BINARYMERGE($TKG_1 = (\mathcal{E}_1, \mathcal{R}_1), TKG_2 = (\mathcal{E}_2, \mathcal{R}_2), \theta_E, \theta_R$)
      // —— *Entity Resolution* —— //
2:    **Initialize** mapping $M \leftarrow \emptyset$
3:    **for all** entity $e \in \mathcal{E}_1$ **do**
4:        **if** there exists $e' \in \mathcal{E}_2$ such that $e.$name $= e'.$name and $e.$label $= e'.$label **then**
5:            $M(e) \leftarrow e'$
6:        **else**
7:            Compute $s \leftarrow \max_{e' \in \mathcal{E}_2} \cos\left(e.\mathbf{v}, e'.\mathbf{v}\right)$ ▷ Cosine similarity of e and e' embeddings
8:            Let $e^* \leftarrow \arg\max_{e' \in \mathcal{E}_2} \cos\left(e.\mathbf{v}, e'.\mathbf{v}\right)$
9:            **if** $s \geq \theta_E$ **then**
10:               $M(e) \leftarrow e^*$
11:            **else**
12:               $M(e) \leftarrow e$
13:            **end if**
14:        **end if**
15:    **end for**
16:    $\mathcal{E}_{\text{merged}} \leftarrow \mathcal{E}_2 \cup \{e \mid M(e) \notin \mathcal{E}_2\}$
      // —— *Relation's Name Resolution* —— //
17:    $\mathcal{R}_1^{\text{updated}} \leftarrow \emptyset$
18:    **for all** relation $r \in \mathcal{R}_1$ **do**
19:        **Update endpoints:** $r.$startEntity $\leftarrow M(r.$startEntity$)$, $r.$endEntity $\leftarrow M(r.$endEntity$)$
20:        Compute $s_r \leftarrow \max_{r' \in \mathcal{R}_2} \cos\left(r.\mathbf{v}, r'.\mathbf{v}\right)$ ▷ Cosine similarity of r and r' names embeddings
21:        Let $r^* \leftarrow \arg\max_{r' \in \mathcal{R}_2} \cos\left(r.\mathbf{v}, r'.\mathbf{v}\right)$
22:        **if** $s_r \geq \theta_R$ **then**
23:            **Update names:** Update $r.$name $\leftarrow r^*.$name
24:        **end if**
      // —— *Temporal Resolution* —— //
25:        **if** there exists $r' \in \mathcal{R}_2$ such that r is similar to r' **then**
        // For similar relations, their times are merged
26:            **Update start time:** $r'.t_{start} \leftarrow r'.t_{start} \cup r.t_{start}$
27:            **Update end time:** $r'.t_{end} \leftarrow r'.t_{end} \cup r.t_{end}$
28:            **Update observation time:** $r'.t_{obs} \leftarrow r'.t_{obs} \cup r.t_{obs}$
29:        **end if**
30:        $\mathcal{R}_1^{\text{updated}} \leftarrow \mathcal{R}_1^{\text{updated}} \cup \{r\}$
31:    **end for**
32:    $\mathcal{R}_{\text{merged}} \leftarrow \mathcal{R}_2 \cup \mathcal{R}_1^{\text{updated}}$
33:    **return** $(\mathcal{E}_{\text{merged}}, \mathcal{R}_{\text{merged}})$
34: **end function**

Atomicity: Compound sentences must be split. Each atomic fact must contain exactly one piece of information. Redundancies and duplicated information were removed.

Decontextualization: Facts must be understandable in isolation. Pronouns (e.g., "he", "it") were replaced with full entity names, and necessary modifiers were included.

Temporal normalization: All relative time references were converted to absolute dates based on the t_{obs}. For example: *Today* → exact t_{obs}; *Yesterday* → $t_{obs} - 1$ day; *Last week* → Monday of the week preceding t_{obs}; *This year* → January 1st of the t_{obs} year.

End actions: If the text indicated the end of a role or action (e.g., "leaving a position"), the termination was captured along with its specific timestamp.

E.2   Phase 2: 5-tuple extraction

In the second phase, the model extracted temporal 5-tuples $(e_s, r, e_o, t_{\text{start}}, t_{\text{end}})$ from the atomic facts generated in Phase 1. Similarly, annotators verified and corrected the extracted tuples for exhaustivity, factual accuracy, and strict temporal validity, ensuring adherence to the following guidelines:

Entity Annotation   Entities were defined as distinct concepts (e.g., Person, Organization, Position). Dates were strictly excluded from entity mentions (they are reserved for temporal arguments).

Relationship and temporal annotation   Relations were extracted to capture the interaction between entities, with strict temporal rules:

- **Canonical present tense:** All relation names

Algorithm A.2 Parallel Merge of TKGs

1: **function** PARALLELMERGE($TKGs$, θ_E, θ_R)
2:     **Input:** A list of temporal knowledge graphs

$$TKGs = \{TKG_1, TKG_2, \ldots, TKG_n\}$$

3:     $current \leftarrow TKGs$
4:     **while** $|current| > 1$ **do**
5:         $mergedResults \leftarrow \emptyset$
6:         Let $n \leftarrow |current|$
7:         **Form pairs:**

$$pairs \leftarrow \{\, (current[2i],\ current[2i+1]) \mid 0 \le i < \lfloor n/2 \rfloor \,\}$$

8:         **if** n is odd **then**
9:             $leftover \leftarrow current[n-1]$
10:        **else**
11:           $leftover \leftarrow$ null
12:        **end if**
13:        **for all** each pair (TKG_a, TKG_b) in $pairs$ **in parallel do**
14:            $merged \leftarrow$ **BINARYMERGE**(TKG_a, TKG_b, θ_E, θ_R)
15:            Add $merged$ to $mergedResults$
16:        **end for**
17:        **if** $leftover \neq$ null **then**
18:            Add $leftover$ to $mergedResults$
19:        **end if**
20:        $current \leftarrow mergedResults$
21:     **end while**
22:     **return** $current[0]$
23: **end function**

---

were annotated in the present tense (e.g., *is_CEO*, *works_at*) regardless of whether the event was past, present, or future. This enhances semantic consistency across the TKG.

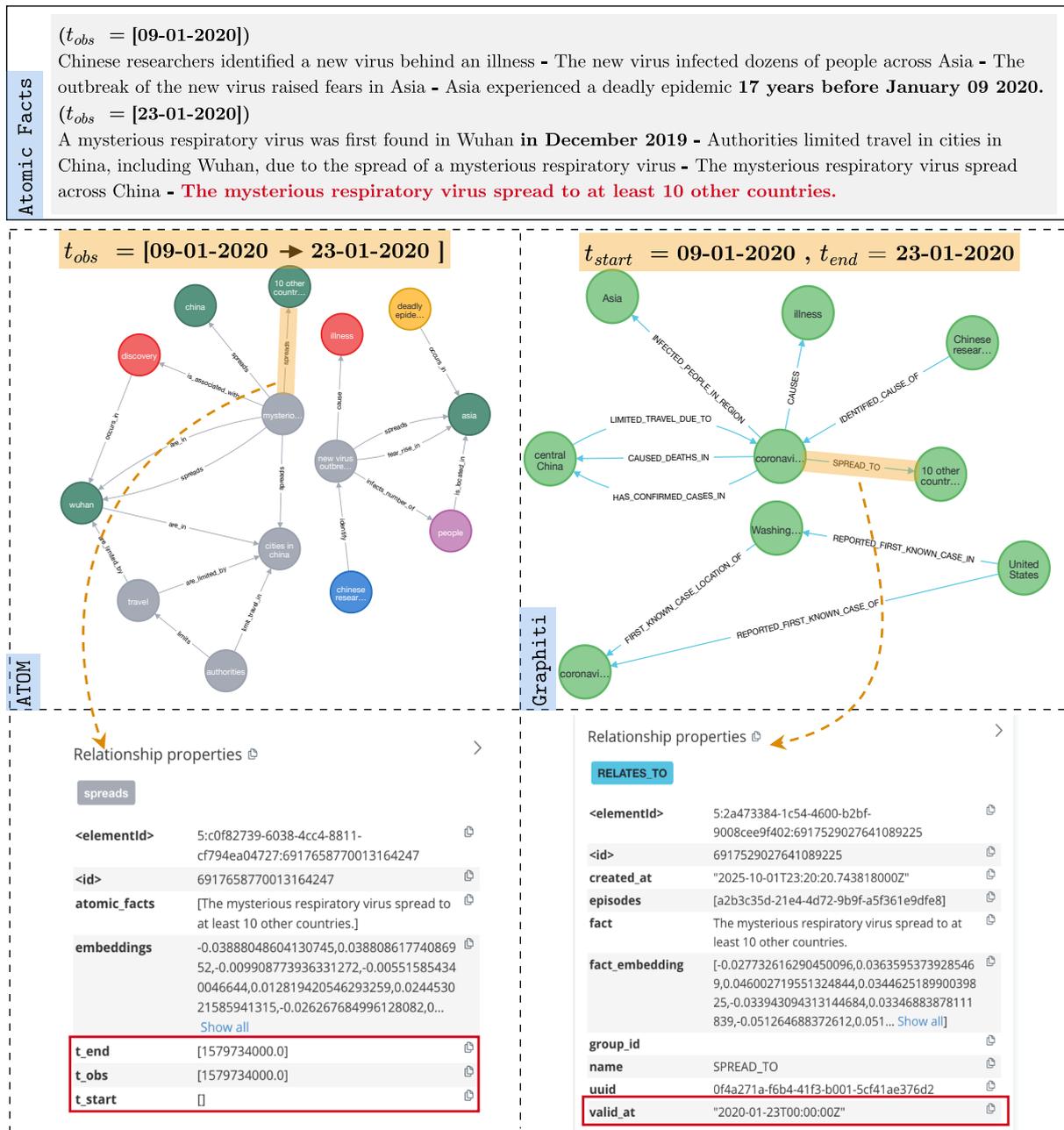- **Validity periods:** Temporal information was mapped strictly to t_{start} (validity start) and t_{end} (validity end) based on the absolute dates resolved in Phase 1.

    - *Affirmative actions* (e.g., "became CEO") populated t_{start}.
    - *End actions* (e.g., "is no longer the CEO") populated t_{end}, while the relation name remained in the affirmative (e.g., *is_CEO*).

Table T.1: 2020-COVID-NYT Statistics Analysis. We use lead paragraphs as they encapsulate the article's key facts, while the full article text is often unavailable in research datasets due to licensing restrictions and would introduce unnecessary verbosity without proportional information gain.

Metric	Value
Basic Dataset Information	
Total Articles	1,076
Grouped Articles (by pub. date)	274
Average Tokens per Group	206 ± 156
Date Range	2020-01-09 to 2020-12-30
Atomic Facts Analysis	
Total atomic facts	4,223
Atomic facts with validity time	2,037
Atomic facts without validity time	2,186
Knowledge Graph Structure	
Total 5-tuples	7,210
Number of atomic TKGs	4,223
Avg number of 5-tuples per atomic TKG	~ 2

Table T.2: Cost and token usage analysis on the 2020-COVID-NYT dataset. **(L)** denotes direct extraction from Lead paragraphs. **(F)** denotes the full ATOM pipeline, broken down into **Module-1** (atomic fact decomposition) and **Module-2** (5-tuple extraction from atomic facts).

	Direct Extraction	ATOM (F)		
Metric	(L)	Module-1	Module-2	Total
Token Usage Statistics				
Total Input Tokens	271,108	237,192	285,618	522,810
Total Output Tokens	83,757	38,646	217,110	255,756
Avg Input per Article	2,222	1,944	2,341	4,285
Avg Output per Article	687	317	1,780	2,096
Estimated Cost (USD)				
claude-sonnet-4-2025-01-31	$1.04	$0.64	$2.06	$2.70
gpt-4.1-2025-04-14	$0.61	$0.40	$1.15	$1.55

Figure F.1: Two DTKGs constructed using ATOM and Graphiti from 09-01-2020 (in UNIX, 1578524400) to 23-01-2020 (in UNIX, 1579734000) from 2020-COVID-NYT dataset. **Left (ATOM):** Preserves observation times (t_{obs}) separately from validity periods, with timestamps encoded in UNIX format to eliminate overhead associated with string parsing operations and timezone conversion calculations. **Right (Graphiti):** Treats observation time as validity start time. $valid\_at$ corresponds to t_{start} in Graphiti's time modeling. The highlighted fact *"The mysterious respiratory virus spread to at least 10 other countries"* is observed on 23-01-2020, but this does not guarantee the spread occurred at that time. ATOM's dual-time modeling prevents such temporal misattribution.
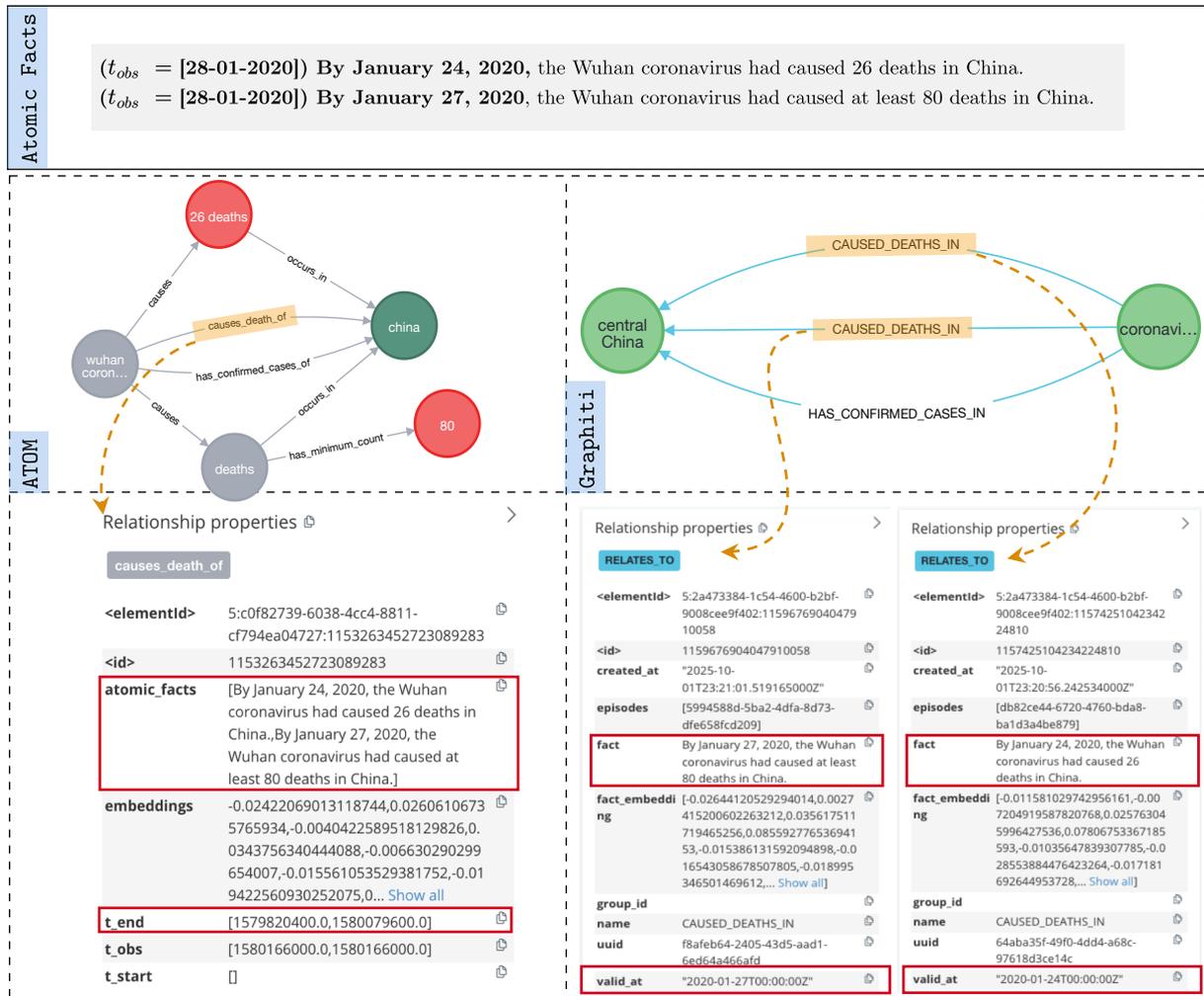
Figure F.2: Temporal resolution comparison between ATOM and Graphiti. Two atomic facts observed on January 28, 2020, report death counts from January 24 (26 deaths) and January 27 (at least 80 deaths). **Left (ATOM)**: performs temporal resolution by detecting similar relations and extending their validity period history (t_{end} in the figure). **Right (Graphiti)**: creates separate relations for each atomic fact, resulting in duplication. Moreover, Graphiti misinterprets *"By January 24, 2020"* and *"By January 27, 2020"* as validity start times rather than validity end times, leading to temporal misattribution.

Figure F.3: Temporal resolution comparison between ATOM and Graphiti. Two atomic facts observed on different dates (April 16 and April 19, 2020) describe protest activities during two time periods (*"the week of April 13"* and *"the week of April 19"*). **Left (ATOM)**: merges similar relations and extend their validity periods (t_{start} and t_{end} in the figure). **Right (Graphiti)**: maintains separate relations for each atomic fact. Moreover, Graphiti fails to translate *"In the week of April 13, 2020"* and *"In the week of April 19, 2020"* into proper validity periods as ATOM does.