Chapter 2

ChronoGrapher: Event-centric Knowledge Graph Construction via Informed Graph Traversal

Inès Blin*†, Ilaria Tiddi†, Remi van Trijp*, Annette ten Teije†

*Sony Computer Science Laboratories - Paris
†Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Abstract

Event-centric knowledge graphs help enhance coherence to otherwise fragmented and overwhelming data by establishing causal and temporal connections using relevant data. We address the challenge of automatically constructing event-centric knowledge graphs from generic ones. We present ChronoGrapher, a two-step system to build an event-centric knowledge graph from grand events such as the French Revolution. First, an informed graph traversal retrieves connected sub-events from large, open-domain knowledge graphs. We define event-centric filters to prune the search space and a heuristic ranking to prioritise nodes like events. Second, we combine a rule-based system and information extraction from text to build event-centric knowledge graphs. ChronoGrapher demonstrates adaptability across datasets like DBpedia and Wikidata, outperforming approaches from the literature. To evaluate the utility of these graphs, we conduct a preliminary user study comparing different prompting techniques for event-centric question-answering. Our results demonstrate that prompts enriched with event-centric knowledge graph triples yield more factual answers compared to those enriched with generic knowledge graph triples or base prompts, achieving groundedness scores of 2.85, 2.24, and 1.11 respectively, while preserving succinctness and relevance.

Keywords Event-centric KGs, Graph traversal, Knowledge graphs

2.1 Introduction

We tackle the problem of building event-centric knowledge graphs (KGs) automatically starting from generic KGs. An event-centric KG can be seen as a KG that focuses on events as central entities, rather than generic entities. An event is an occurrence or happening that can be described by its attributes and relationships to other entities within a knowledge graph. It includes information such as time, location, participants, and other relevant properties. Relating events together provide coherence to otherwise fragmented and overwhelming data by

establishing causal and temporal connections using relevant data. There are numerous scenarios in which systems could assist individuals in constructing event-centric KGs in their tasks: in news article retrieval for event-centric exploration [76], in the legal domain where experts must associate events that could bolster their case [15], or in the historical domain when generating hypotheses by retrieving valuable information from archives [50].
Let us imagine a historian seeking to understand the events that happened at the end of the French Revolution. Our system aims to produce an event-centric KG, as illustrated in Figure 2.1, which we manually created using mainly the SEM [75] ontology and Wikidata. Figure 2.1 shows that the *Coup of 18 Brumaire directlyPrecedes* the *French Consulate*, and that Napoleon was a *participant* in the *Coup of 18 Brumaire*, and *First Consul* during the *French Consulate*. We can therefore infer that the *Coup of 18 Brumaire* marked the beginning of the *French Consulate* and enabled *Napoleon* to become *First Consul*.

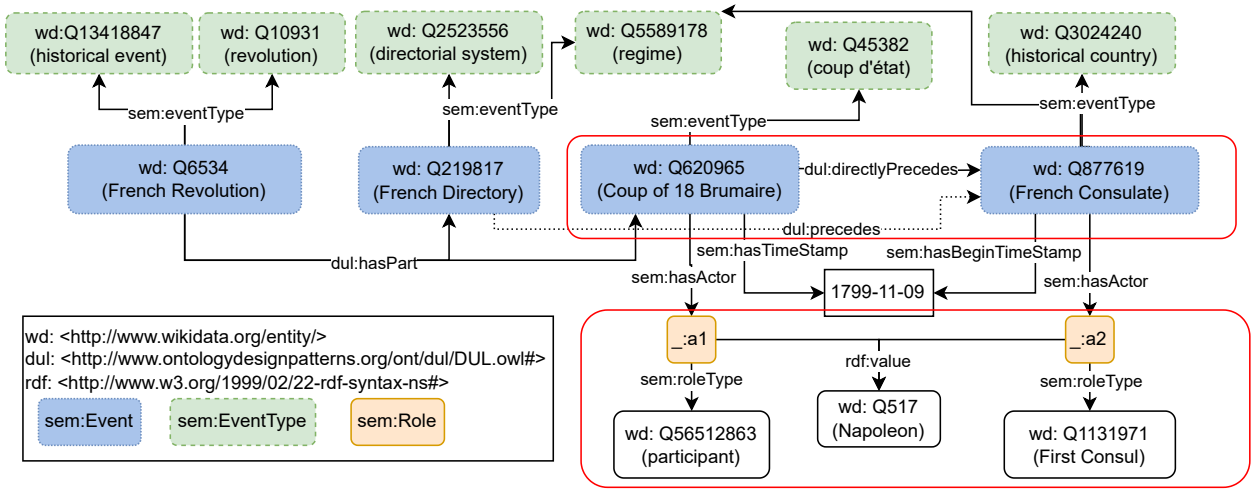

Figure 2.1: One example of an event-centric KG on the Coup of 18 Brumaire.

Building an event-centric KG can be seen as a 2-step process: searching cues in a massive memory (step-1), and analysing and reasoning about these cues (step-2) [67]. KGs have been used as background knowledge to support the construction of event-centric KGs [23]. In this context, these two steps can be translated into identifying relevant content from a generic KG (step-1), and linking this content with additional information to create an event-centric KG (step-2). Most existing event-centric KG resources are however built from annotated text [42, 53, 28] or from generic KGs [20] (step-2), but do *not* focus on identifying relevant events for event-centric KG construction (step-1). Likewise, event subgraph extraction (step-1) has been used for downstream tasks such as question-answering (QA) [68, 35, 44], but does *not* focus on building event-centric KGs (step-2). We propose ChronoGrapher that both retrieves relevant events and generates an event-centric KG, as well as an evaluation of an event-centric KG constructed by ChronoGrapher in a QA setting in the form of a user study. We make our code openly available[1].

Our objective is to extract sequences of events as a plausible answer for a question, such as summarising the French Revolution, or discussing whether Napoleon and Paul Barras were both involved in the Coup of 18 Brumaire. We define three research questions: **(RQ1)** *How to extract relevant events from a generic KG? (step-1)* **(RQ2)** *How to create an event-centric KG from these events? (step-2)* **(RQ3)** *How to evaluate the utility of the event-centric KG*

[1] https://github.com/SonyCSLParis/graph_search_framework/tree/main

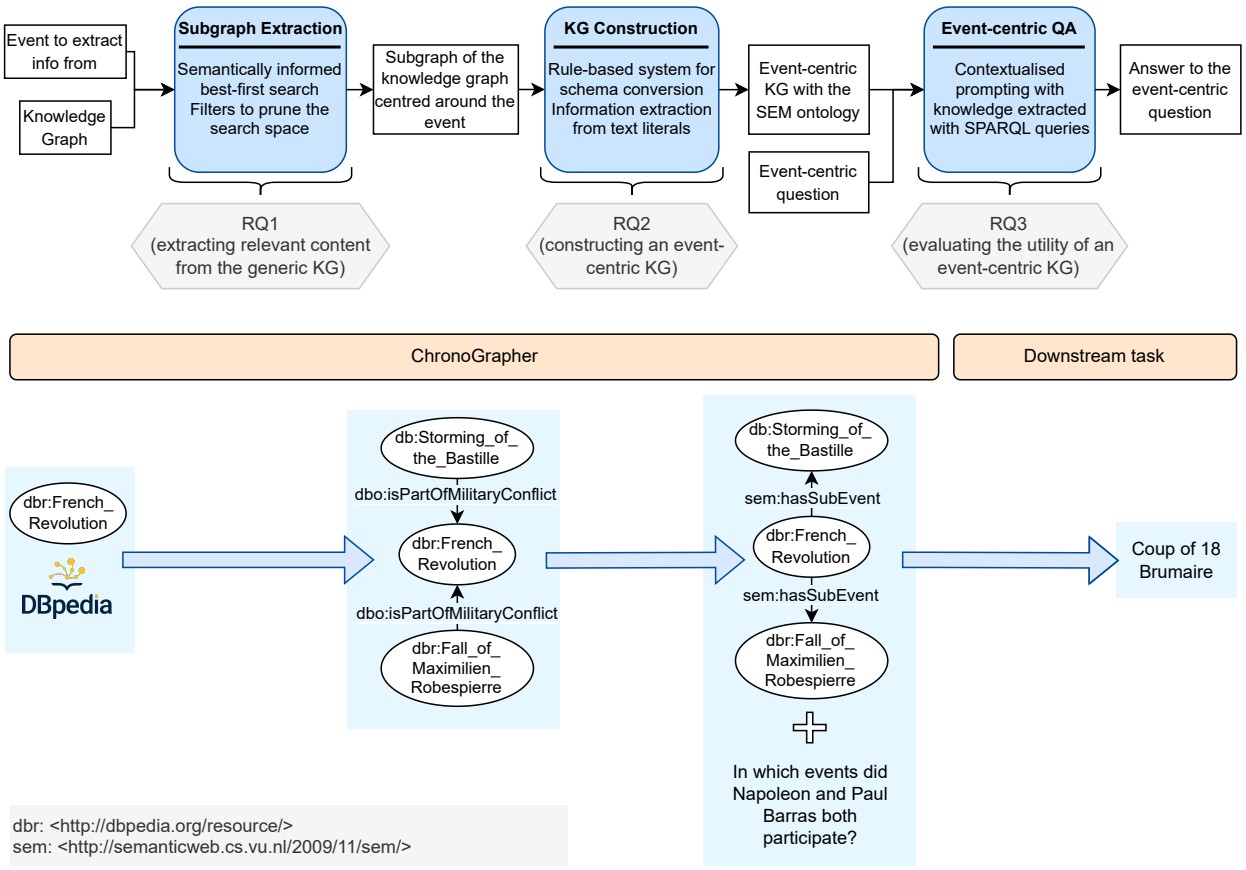

Figure 2.2: Overview of the diverse tasks we tackle to answer our research questions. The example on the bottom is with the French Revolution as starting node and DBpedia as the KG. ChronoGrapher integrates both a subgraph extraction step through a semantically informed best-first search, and a KG construction step through a rule-based system and information extraction from text. The downstream task is event-centric question-answering.

constructed by ChronoGrapher? Figure 2.2 provides an overview of the different tasks we tackle, as well as the input and output of each components.

For **RQ1**, ChronoGrapher contains an informed graph traversal to retrieve connected sub-events from large, open-domain KGs. It is a *semantically-informed* best-first search. First, it uses *event-centric filters* to prune the search space and to guide the exploration of the graph, and it identifies which ones are the most relevant to build an event-centric KG. Second, it adds a ranking step to prioritise nodes, which, to the best of our knowledge, has not been used in search systems starting from one node in KGs. We define the ranking heuristics based upon syntactic and semantic properties of the graph.

For **RQ2**, ChronoGrapher combines a rule-based system on triples with information extraction from text to generate an event-centric KG. The rule-based systems converts the triples from the original KG into event-centric ones. Furthermore, since the KGs can contain valuable information in text format, we include information extraction from text. We evaluate ChronoGrapher and adapted methods from the literature [33, 18] against ground truth events from EventKG [20]. ChronoGrapher shows adaptability and efficiency across datasets like DBpedia and Wikidata, outperforming existing methods. Concerning the technical novelty of the work, methods for representing events in KGs are underdeveloped, manually built and time-consuming, and ChronoGrapher is a method for building event-centric KGs automatically from a generic KG. Unlike most approaches that relied on manual efforts or solely on text extraction, ChronoG-

rapher integrates data from existing KGs and texts, and results in more concise and precise outputs than baselines adapted from existing systems. The modularity of our approach ensures flexibility and adaptability, while the novel combination and adaptation of previously unintegrated elements contributes to the field and improves the adapted baselines [33, 18].
For **RQ3**, we evaluate the event-centric KGs built with ChronoGrapher via a QA downstream task. A human evaluation is conducted to compare three types of zero-shot QA prompts (without explicit context, with triples from generic KGs, with triples from event-centric KGs) and six types of different questions. Our results demonstrate that prompts enriched with event-centric knowledge graph triples yield more factual answers compared to those enriched with generic knowledge graph triples or base prompts, achieving groundedness scores of 2.85, 2.24, and 1.11 respectively, while preserving succinctness and relevance.

2.2 Related Work

New approaches have emerged to build event-centric KGs with various applications (i). To the best of our knowledge, building event-centric KGs *from generic KGs automatically* is novel. We therefore adapt and improve existing work to harness the potential of event-centric KGs (ii). To assess the usefulness of event-centric KGs, we lastly present works that used KGs to improve the quality of answers generated by large language models (LLMs) (iii).

(i) Event-centric KGs. Recent research on KGs emphasises the need to have more event-centric structures to facilitate the analysis of sequence of events [61]. Event-centric KGs were built in various domains: news [53, 28, 42], political events [5], novels [38], the biomedical domain [47], tourism [79], the legal domain [15], olfactory heritage [45] and plenary debates [31]. More particularly in the digital humanities and cultural heritage domain, knowledge graph resources are emerging to link objects to their events and underlying stories. This is for example the case for Arco, an Italian cultural heritage KG [6], that was built on XML date using the eXtreme Design methodology [4]. Another application is to better understand history, may it be through the history of a city like Amsterdam [54] or through historical archives [65, 74, 72, 51]. [74] focused on event extraction and modelling from text archive, and [72] proposes an agile cataloguing process to build a knowledge graph. The work that is the most closely related to ours is EventKG [13], an event-centric KG that was built manually starting from large, generic KGs, such as DBpedia or Wikidata.
We attempt to build EventKG automatically, and use this dataset as golden standard in our evaluation (see Section 2.4.2). The above resources extracted events without a focus on identifying relevant events for an event-centric KG. Our approach targets the extraction of relevant events locally.

There are different applications of using narrative graph representations. [25] worked on enhancing sequential pattern mining with time and reasoning, to analyse complex dynamical systems. Understanding such systems can help to acquire new knowledge but also to predict and control its behaviour. Event-centric representations can also help in decision-making, for instance in the epidemiological domain [3] or maritime transportation [12]. [13] lastly uses structured narrative representations to build hypotheses over knowledge bases.

(ii) Methods for Event-centric KG construction. We present different research domains that are relevant to building event-centric KGs: linked-traversal based methods, graph navigational languages and entity relatedness and summarisation. We also compare it to ChronoGrapher.
Linked-traversal based methods identify and extract the most relevant set of information

for a given task. One of the pioneering system in this domain is LDSpider [33], which employs diverse search strategies such as breadth-first search or load-balancing to crawl web data. Subsequent systems aimed to reduce the search space during the traversal, encoding features like property selection [64] or basic graph patterns [73]. Systems like Linked Data-fu [69] and LiDAQ [73] added reasoning capabilities to their approaches. Other systems like SQUIN [27] focused on optimising query execution efficiency.

Graph navigational languages are used to explore Linked Data iteratively by providing instructions to retrieve nodes or subgraphs. There are two types of approaches, namely SPARQL extensions [17, 26, 60] and standalone languages [18, 28]. Most of these approaches use regular expressions, like SPARQLeR [39] and PSPARQL [1], and later nested regular expressions, like [80] and nSPARQL [56] to explore the graph. Although our approach is more similar to a link traversal-based method, we compare the scope and output of our method to those of navigational languages. Our approach can handle local graphs and any SPARQL endpoint, and the output of our method is regions, similar to GeL [16] and NautiLOD [18]. The most expressive system appears to be LDQL [28] that separates the pattern processing and the source selection.

Entity Relatedness systems aim to determine the optimal path connecting two nodes in a KG [36]. It involves computing all paths with search heuristics and activation criteria to prune the search space and ranking paths [31]. Search heuristics include bidirectional search [13, 71], backward search heuristic [30], SPARQL queries [59], exploratory association search [10], and A* [11]. Activation criteria include entity clustering [10], metrics based on node degree [13, 52] or ontology [48], and similarity measures [30, 11] like Jaccard, Wikipedia link-based measures [78] or SimRank [34]. For ranking paths, metrics include PF-ITF [59], exclusivity-based relatedness [30] and point mutual information [32]. Table 2.1 provides an overview of the entity relatedness systems we described. **Entity Summarization** systems [46] take a KG as input and output a graph that keeps certain original properties of the graph. The methods used to select nodes are similar to the ranking ones from entity relatedness systems.

Table 2.1: Overview of entity relatedness systems.

System	Path Search	Activation Criteria	Ranking
[71]	Bidirectional search	Primitives	Cost functions
Profiler [30]	Backward search heuristic	Similarity measures	N/A
RECAP [59]	SPARQL queries	N/A	Path informativeness
EXPLASS [10]	Exploratory association search	Clustering entities	N/A
[11]	A*	Jaccard-based metrics	Top k paths
[52]	Shortest paths	Low-degree nodes	Top k paths
REX [13]	Bidirectional search	Degree of node	Interestingness measure
RelFinder [48]	All paths	Class, link and length filter	N/A

Our method traverses KGs to extract relevant events for event-centric KG building, akin to link traversal-based methods and graph navigational languages. It handles local graphs from any SPARQL endpoint, producing subgraphs (like [16] and [18]). Addressing limitations in expressiveness and navigation control of link-traversal methods, our approach provides enhanced control and reasoning. It employs filters for search space pruning without prior knowledge of predicates, unlike graph navigational languages requiring regular expressions. Additionally, we introduce a ranking step inspired by entity relatedness and summarization systems, prioritizing nodes based on a cost function. To the best of our knowledge, this feature is absent in existing navigational systems. Unlike entity relatedness systems that pre-compute all paths between entities, our system dynamically computes paths. Furthermore, our approach requires only one starting node as input, with target nodes defined as variables. It extracts subgraphs from

the original KG, aiming to retrieve all events of a given entity, disregarding specific property considerations as in entity summarization systems.

(iii) Retrieval Augmented Generation (RAG). Studies have shown that LLMs store factual knowledge in their parameters [58], e.g. in the form of implicit knowledge bases [62]. However, the mechanisms to retrieve such knowledge remain opaque. RAG systems [9] aim to provide a more interpretable way to access knowledge by combining a generative model and an explicit *knowledge retriever component*. Such systems tend to generate more specific, diverse and factual language [43], and have proven beneficial for downstream tasks such as QA [8] or reasoning [70]. Systems like ORQA [41] or REALM [24] learn the retriever and the reader jointly, whereas systems like DrQA [8] have a fixed document retrieval component. [37] proposes a dense passage retriever to index all documents in a latent space. [57] augments the reader with a retriever in an unsupervised way.
In our user study, we compare three types of zero-shot prompting for QA: one prompting without explicit context, and two prompting with triples from generic KGs and event-centric KGs respectively. Our *knowledge retriever component* is based on `SPARQL` queries, to retrieve triples from a knowledge base.

2.3 ChronoGrapher

As displayed in Figure 2.2 from Section 2.1, ChronoGrapher takes an event IRI as input, searches all sub-events of that event, and outputs an event-centric KG. It is flexible and can be applied to retrieve any type of node, such as people or places, and to any input KG. We present preliminary content (Section 2.3.1), the semantically-informed graph traversal method to extract relevant events (RQ1) (Section 2.3.2) and the methods to build an event-centric KG (RQ2) (Section 2.3.3).

2.3.1 Preliminaries

We first introduce the generic concepts of knowledge graph, event-centric knowledge graph and event.

[Knowledge Graph] A **knowledge graph** KG is an RDF graph where the nodes represent entities (subjects or objects) and literals (objects) and the edges represent the relationships (predicates) between these entities and literals. Formally, $\mathcal{KG} = \{(s, p, o) | s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities (nodes) in the KG, $\mathcal{L}$ is the set of literals in the KG, and $\mathcal{R}$ is the set of predicates (edges) representing relationships in the KG.

[Event-centric KG and Event] In this work, an **event-centric knowledge graph** is a KG whose vocabulary, classes and relationships originate from the SEM [75] and the NIF [18] ontologies. There are four core classes in SEM: `sem:Event` (what), `sem:Actor` (who), `sem:Place` (where) and `sem:Time` (when). The NIF ontology enables to integrate text data in KGs. An **event** is furthermore defined as an occurrence in time that can be characterised by attributes from the SEM ontology, such as actor and location.

We second formally define concepts that are useful for our link traversal-based method, before presenting ChronoGrapher into details: the ingoing and outgoing triples of a node in a KG, node expansion, best-first search algorithm, event retrieval path and relation patterns. Please note that these definitions are generic, and we later instantiate and use them to build the event-centric KGs (cf. Section 2.4.1 for the experimental settings).

[Ingoing triples] Let $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$, and $\mathcal{P}(\mathcal{KG}) = \{\mathcal{A}|\mathcal{A} \subseteq \mathcal{KG}\}$ the set of all subsets of $\mathcal{KG}$. The **ingoing triples** of a node n are formally defined as:

$$\begin{aligned} ingoing_{\mathcal{KG}} &: \mathcal{E} \longrightarrow \mathcal{P}(\mathcal{KG}) \\ & n \longmapsto \{(s,p,n) \mid \exists p \in \mathcal{R},\ \exists s \in \mathcal{E},\ \text{ such that } (s,p,n) \in \mathcal{KG}\} \end{aligned}$$

[Outgoing triples] Let $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$, and $\mathcal{P}(\mathcal{KG}) = \{\mathcal{A}|\mathcal{A} \subseteq \mathcal{KG}\}$ the set of all subsets of $\mathcal{KG}$. The **outgoing triples** of a node n are formally defined as:

$$\begin{aligned} outgoing_{\mathcal{KG}} &: \mathcal{E} \longrightarrow \mathcal{P}(\mathcal{KG}) \\ & n \longmapsto \{(n,p,o) \mid \exists p \in \mathcal{R},\ \exists o \in \mathcal{E} \cup \mathcal{L},\ \text{ such that } (n,p,o) \in \mathcal{KG}\} \end{aligned}$$

[Node Expansion] **Expanding a node** n means retrieving its ingoing and outgoing triples.

[Best-first search algorithm] A **best-first search algorithm** expands unvisited nodes with the highest scores based on a predefined heuristic.

[Event-retrieval task] Given $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$ and an event s, the aim of the **event retrieval task** is to traverse $\mathcal{KG}$ starting from s to retrieve the sub-events of s, $e = \{e_1, ..., e_n\}$. For evaluation, e is compared to ground truth sub-events.

[Relation Patterns and Node Satisfaction] Let $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$. We define the following relation patterns in $\mathcal{KG}$ as Boolean functions:

- **Single-step outgoing predicate pattern**: Let a predicate $p \in \mathcal{R}$, we formally define $opp_{p,\mathcal{KG}}$ as follows:

$$\begin{aligned} opp_{p,\mathcal{KG}} &: \mathcal{E} \longrightarrow \{0,1\} \\ & n \longmapsto \begin{cases} 1 & \text{if } \exists\, o \in \mathcal{E} \text{ such that } (n,p,o) \in \mathcal{KG}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- **Single-step ingoing predicate pattern**: Let a predicate $p \in \mathcal{R}$, we formally define $ipp_{p,\mathcal{KG}}$ as follows:

$$\begin{aligned} ipp_{p,\mathcal{KG}} &: \mathcal{E} \longrightarrow \{0,1\} \\ & n \longmapsto \begin{cases} 1 & \text{if } \exists\, s \in \mathcal{E} \text{ such that } (s,p,n) \in \mathcal{KG}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- **Single-step outgoing predicate-object pattern**: Let a predicate $p \in \mathcal{R}$ and an entity $e \in \mathcal{E}$, we formally define $opop_{p,e,\mathcal{KG}}$ as follows:

$$\begin{aligned} opop_{p,e,\mathcal{KG}} &: \mathcal{E} \longrightarrow \{0,1\} \\ & n \longmapsto \begin{cases} 1 & \text{if } (n,p,e) \in \mathcal{KG}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- **Single-step ingoing predicate-object pattern**: Let a predicate $p \in \mathcal{R}$ and an entity $e \in \mathcal{E}$, we formally define $ipop_{p,e,\mathcal{KG}}$ as follows:

$$\begin{aligned} ipop_{p,e,\mathcal{KG}} &: \mathcal{E} \longrightarrow \{0,1\} \\ & n \longmapsto \begin{cases} 1 & \text{if } (e,p,n) \in \mathcal{KG}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

For each pattern defined above, a node $n \in \mathcal{E}$ **satisfies** this pattern if its image is 1.

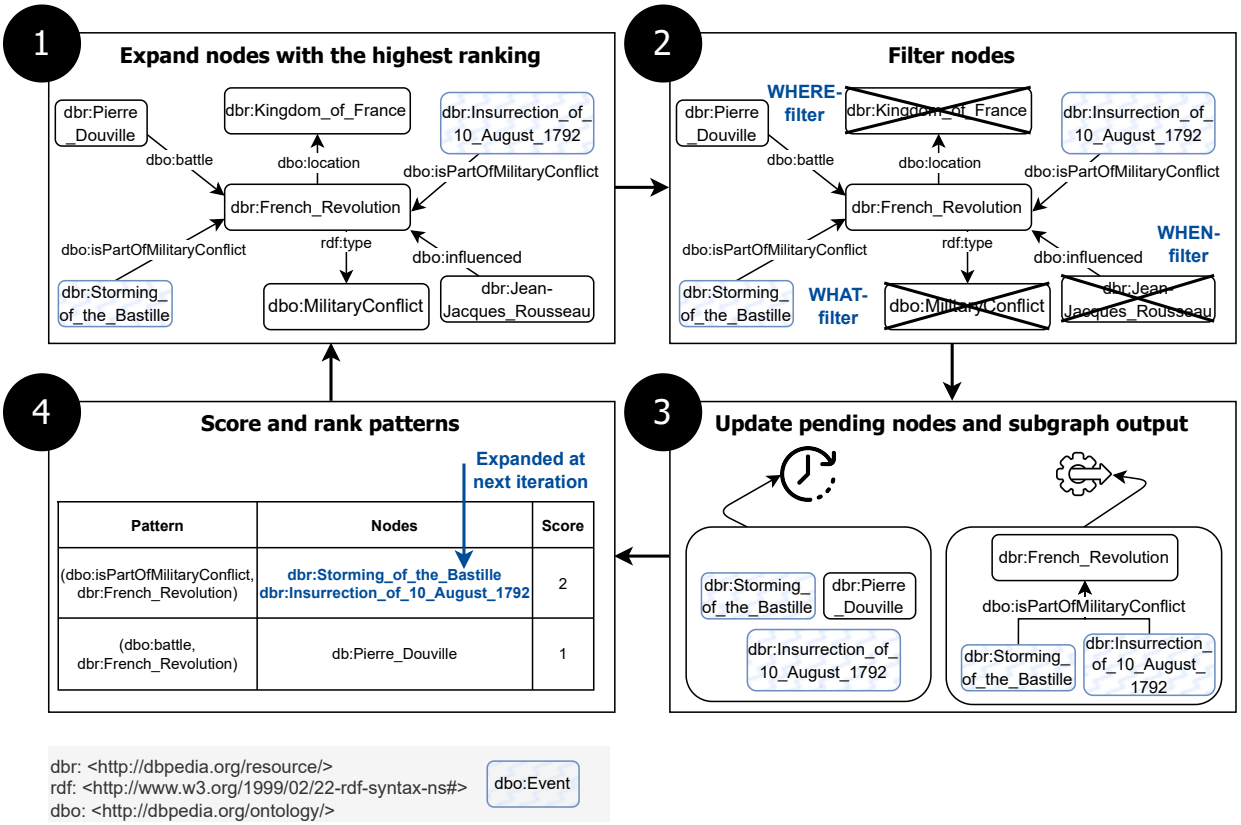

Figure 2.3: Description of one iteration during the informed graph traversal. For clarification, we denote as $\mathcal{DB}$ the set of triples in DBpedia. The type of ranking is predicate-object frequency. The *WHERE*, *WHAT* and *WHEN* filters are activated (formally defined as $e_filter_{\{dbo:Location\},\mathcal{DB},rdf:type}$, $p_filter_{rdf:type}$ and $t_filter_{dbo:startDate,1789-05-05,dbo:endDate,1799-12-31}$ respectively).

2.3.2 Relevant Content Extraction (RQ1)

The first component of ChronoGrapher is a modified best-first search that extracts relevant content to build an event-centric KG. We define generic filters to prune the search space, and instantiate them as event-centric filters for the experimental settings (Section 2.4.1). We use a combination of ranking metrics and preference function. As illustrated in Figure 2.3[2], each iteration in the graph search process is composed of four stages. During Stage 1, nodes with the highest ranking are expanded, ie. their ingoing and outgoing nodes are retrieved. During Stage 2, we apply filters to remove newly visited ingoing and outgoing nodes from the search space. During Stage 3, we update the ingoing and outgoing nodes that were not discarded and the subgraph output. During Stage 4, we score and rank patterns. Both filters and ranking are detailed later in this Section.

Algorithm 2.3.2 presents our four-step method: (1) nodes with the highest ranking are explored in the graph, (2) a filtering step using event-centric filters removes newly visited nodes from the search space, (3) the pending nodes and subgraph output are updated in the third step, (4) nodes in the pending set are scored and ranked, and those with the highest score are selected for the next iteration. *nodes* is initialised in line 2 of the Algorithm ; it is the set of nodes that will be expanded at the next iteration and is considered the priority queue of the pending set of nodes $\mathcal{P}$ initialised in line 1.

[2]Images taken from `https://www.flaticon.com/free-icons/watch` and `https://www.flaticon.com/free-icons/output`

input: $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$: knowledge graph to traverse,
n_{start}, t_{start}, t_{end}: node to start the search, its start and end dates n: number of iteration,
$to_extract$: type of nodes to extract ,
$filtering$, $preferencefunction$, $scoring$: defined in Section 2.3.
output: $e = \{e_1, ..., e_n\}$: events extracted with the traversal,
[1] initialise $\mathcal{P}$: list of pending nodes that could be expanded
initialise $nodes = \{n_{start}\}$: list of nodes to be expanded
[h!] initialise $\mathcal{N}_{expanded}$: nodes visited throughout the traversal
initialise e: events extracted with the traversal $i \in \{1, ..., n\}$ $node \in nodes$ add $node$ in $\mathcal{N}_{expanded}$
$ingoing = Ingoing(node, \mathcal{KG})$ Expanding
$outgoing = Outgoing(node, \mathcal{KG})$ Expanding
$to_add_output, pending = Filter(ingoing, outgoing)$ Filtering
add $pending$ to $\mathcal{P}$, to_add_output to e Updating $nodes = selectNodesToExpand(\mathcal{P})$ Scoring and Preference Function
return e

Event-centric filters.

We introduce the filters to prune the search space in Definition 2.3.2. The definitions of the filters are generic, but the examples we give and the filters we use in our experiments are event-centric because we define them following the SEM [75] ontology, further detailed in Section 2.3.1. Each of the filters is defined according to one of the four core classes of SEM. These filters could be implemented using filters in graph navigational languages like NautiLOD [18], or using SPARQL queries. However, the latter methods do not permit to rank patterns. In step 2 of Figure 2.3, filters function dynamically. Expanding the search to other KGs only requires identifying predicates for temporal data and classes for locations or people. We further detail the filters' definitions.

[Event-centric filters] We hereafter define three types of event-centric filters: predicate-based, entity-based and temporal-based. Each filter is a boolean function that enables to prune the search space in a knowledge graph $\mathcal{KG} = \{(s,p,o)|s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$.

Predicate-based filter.
Let $\mathcal{R}' \subset \mathcal{R}$ and $(s,p,o) \in \mathcal{KG}$. The predicate-based filter discards (s,p,o) from the search space if $p \in \mathcal{R}'$. It can be formally defined as:

$$p_filter_{\mathcal{R}'} : \mathcal{KG} \longrightarrow \{0,1\}$$
$$(s,p,o) \longmapsto \begin{cases} 1 & \text{if } p \in \mathcal{R}' \\ 0 & \text{else} \end{cases}$$

- **WHAT-filter.** For example, $\mathcal{R}' = \{\text{rdf:type}\}$.

Entity-based filter.
Let $\mathcal{E}' \subset \mathcal{E}$, $p \in \mathcal{R}$, $s \in \mathcal{E}$. The entity-based filter discards s from the search space if s is of any type in $\mathcal{E}'$. It can be formally defined as:

$$e_filter_{\mathcal{E}',\mathcal{KG},p} : \mathcal{E} \longrightarrow \{0,1\}$$
$$s \longmapsto \begin{cases} 1 & \text{if } \exists t \in \mathcal{E}', \text{ such that } (s,p,t) \in \mathcal{KG} \\ 0 & \text{else} \end{cases}$$

- **WHO-filter.** For example, $\mathcal{E}' = \{\text{dbo:Person}\}$, $p = \text{rdf:type}$.
- **WHERE-filter.** For example, $\mathcal{E}' = \{\text{dbo:Location}\}$, $p = \text{rdf:type}$.

Temporal-based filter.
Let $\mathcal{L}_{date} \subseteq \mathcal{L}$ be the subset of literals corresponding to dates. Let $(ps, pe) \in \mathcal{R}^2$, $(ts, te) \in \mathcal{L}^2_{date}$. ps and pe are predicates denoting the start and end dates of an events, and ts and te are literal timestamps. The temporal-based filter discards $s \in \mathcal{E}$ from the search space if it ends before ts or if it starts after te. It can be formally defined as:

$$t_filter_{ps,ls,pe,le} \colon \mathcal{E} \longrightarrow \{0,1\}$$
$$s \longmapsto \begin{cases} 1 & \text{if } \exists\, t_1 \in \mathcal{L}_{date}, \text{ such that } (s, ps, t_1) \in \mathcal{KG} \text{ and } t_1 > le \\ & \quad \text{or } \exists\, t_2 \in \mathcal{L}_{date}, \text{ such that } (s, pe, t_2) \in \mathcal{KG} \text{ and } t_2 < ls, \\ 0 & \text{otherwise.} \end{cases}$$

- **WHEN-filter.** For example, ps = dbo:birthDate, pe = dbo:deathDate, ts = "1789-05-05", te = "1799-12-31".

The above filters are parameters of the traversal, and can be activated on demand. In Figure 2.3, the filters we activated are WHAT, WHERE, and WHEN (formally defined as $e_filter_{\{dbo:Location\},\mathcal{DB},rdf:type}$, $p_filter_{rdf:type}$ and $t_filter_{dbo:startDate,1789-05-05,dbo:endDate,1799-12-31}$ respectively). For WHAT, `dbo:MilitaryConflict` is discarded from the search space, since it is the object of the triple `(dbr:French_Revolution,rdf:type,dbo:MilitaryConflict)`. Regarding WHERE, `Kingdom_of_France` is discarded since it is of type `dbo:Location`. Lastly, `Jean-Jacques_Rousseau` is discarded because Rousseau died before the French Revolution started, hence applying the WHEN filter.

Scoring and Preference Function.

This step ranks patterns, and selects the nodes to expand. For the highest scored pattern, all nodes that satisfy the same pattern in the graph are expanded. If there is no remaining pattern and nodes to be expanded, the search stops. There are two core elements: scoring and preference function.

[Scoring] Let $\mathcal{G}'$ a subset of $\mathcal{KG}$.
We first define scoring metrics for ingoing and outgoing predicate patterns. Let p, $p \in \mathcal{R}$, and the patterns ipp_p and opp_p from Definition 2.3.1.

- Predicate Frequency *(pf)*. Edges with predicates that are often used in the dataset will be favoured.
$$pf(ipp_p) = |\{s \mid \exists o \in \mathcal{E}, \text{ such that } (s, p, o) \in \mathcal{G}'\}| \tag{2.1}$$
$$pf(opp_p) = |\{o \mid \exists s \in \mathcal{E}, \text{ such that } (s, p, o) \in \mathcal{G}'\}| \tag{2.2}$$

- Entropy Predicate Frequency *(epf)*. Edges with the most informative predicates in the dataset will be favoured. Let $pp \in \{ipp_p, opp_p\}$.
$$epf(pp) = -\frac{pf(pp)}{|\mathcal{G}'|} \times \log(\frac{pf(pp)}{|\mathcal{G}'|}) \tag{2.3}$$

- Inverse Predicate Frequency *(ipf)*. Edges with predicates that are rarely used in the dataset will be favoured. Let $pp \in \{ipp_p, opp_p\}$.
$$ipf(pp) = \frac{1}{pf(pp)} \tag{2.4}$$

We second define scoring metrics for ingoing and outgoing predicate-object patterns. Let p, $p \in \mathcal{R}$, e, $e \in \mathcal{E}$, and the patterns $ipop_{p,e}$ and $opop_{p,e}$ from Definition 2.3.1.

- Predicate-Object Frequency *(pof)*. This is similar to *pf*, but this metrics also differentiates between the objects.

$$pof(ipop_{p,e}) = |\{s \mid (s,p,e) \in \mathcal{G}'\}| \tag{2.5}$$

$$pof(opop_{p,e}) = |\{o \mid (e,p,o) \in \mathcal{G}'\}| \tag{2.6}$$

- Entropy Predicate-Object Frequency *(epof)*. This is similar to *epf*, but this metrics also differentiates between the objects. Let $pop \in \{ipop_{p,e}, opop_{p,e}\}$.

$$epof(pop) = -\frac{pof(pop)}{|\mathcal{G}'|} \times \log(\frac{pof(pop)}{|\mathcal{G}'|}) \tag{2.7}$$

- Inverse Predicate-Object Frequency *(ipof)*. This is similar to *ipf*, but this metrics also differentiates between the objects. Let $pop \in \{ipop_{p,e}, opop_{p,e}\}$.

$$ipof(pop) = \frac{1}{pof(pop)} \tag{2.8}$$

The scoring metric used in Figure 2.3 is *pof*. The score of the pattern $ipop_{\text{dbo:isPartOfMilitaryConflict,dbr:French_Revolution}}$ is 2 since there are two ingoing nodes of the `dbr:French_Revolution` that are connected to the latter with the predicate `dbo:isPartOfMilitaryConflict`. If the scoring metric was *epof*, the score would be $-\frac{2}{3} \times \log\frac{2}{3} \approx 0.12$.

[Preference Function] The preference function is a boolean function that prioritises a pattern if it meets certain conditions. Let $\mathcal{KG} = \{(s,p,o) | s \in \mathcal{E}, o \in \mathcal{E} \cup \mathcal{L}, p \in \mathcal{R}\}$ a KG. Let $d \in \mathcal{R}$ and $sc \in \mathcal{R}$ denoting predicates for domain and subclass information. Let $t_e \in \mathcal{E}$ a type of events to retrieve. Let $p \in \mathcal{R}$, $e \in \mathcal{E}$ and a pattern pp, $pp \in \{ipp_p, opp_p, ipop_{p,e}, opop_{p,e}\}$. We formally define:

$$pref_fpp_{d,sc,t_e} \colon \{ipp_p, opp_p, ipop_{p,e}, opop_{p,e}\} \longrightarrow \{0,1\}$$

$$pp \longmapsto \begin{cases} 1 & \text{if } \exists\, n \in \mathcal{E}, \text{ such that } (p,d,n) \in \mathcal{KG} \\ & \text{and} (n, sc, t_e) \in \mathcal{KG} \\ 0 & \text{otherwise.} \end{cases}$$

- For example, d = rdfs:domain, sc = rdfs:subClassOf, t_e = dbo:Event.

The preference function prioritises predicates with the highest scores from the *scoring* step, favouring nodes with the correct types. We use the ontological equivalent predicates of `rdfs:domain` and `rdfs:range`. In DBpedia, we have `(dbo:isPartOfMilitaryConflict,rdfs:domain,dbo:MilitaryConflict)` and `(dbo: MilitaryConflict, rdfs:subClassOf*, dbo:Event)`. Consequently, in `(s,dbo:isPartOfMilitaryConflict,o)`, s must be of type `dbo:Event`. With the preference function, the system prefers patterns with `dbo:isPartOfMilitaryConflict` over `dbo:battle` since the former has `dbo:MilitaryConflict` as its domain, a subclass of event type, while the latter lacks explicit information about its domain or range. In Figure 2.3, the preference function does not affect the selection of nodes to expand. Without the preference function, patterns are ranked in descending order of scores from the *scoring* step.

2.3.3 Event-centric KG Population (RQ2)

The second component of ChronoGrapher builds event-centric KGs by combining a rule-based system with information extraction from text. The rule-based system converts the triples into SEM triples based on the predicate labels, aiming to reproduce the KGs from EventKG [20]. An example can be seen in Figure 2.4. On the left side of the figure there is a subgraph taken from the Wikidata page of the Coup of 18 Brumaire (`wd:Q620965`). On the right side of the figure, there is event-centric KG following the SEM ontology.

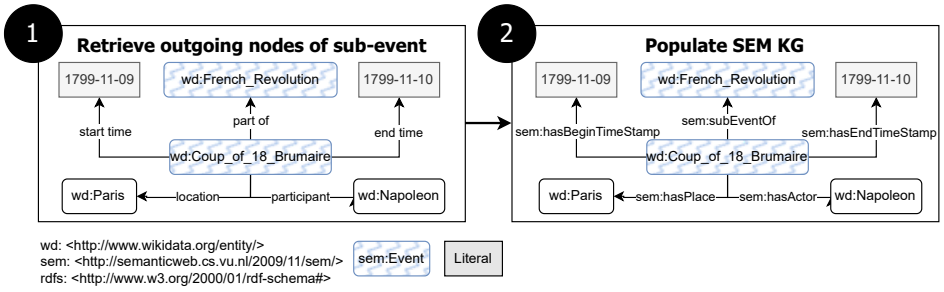

Figure 2.4: Populating a SEM KG from the Coup of 18 Brumaire in Wikidata. *wd* is a prefix used in Wikidata for `https://www.wikidata.org/wiki/`.

Table 2.2 shows the labels that were used to retrieve information from triples. If a predicate contained any of the labels, it would add information in the graph based on its narrative dimension and SEM predicate. As an example, let us take an input triple `(s, p, o)`. If the string of `p` contains the substring *"person"*, then a triple `(s, sem:hasActor, o)` is added to the output graph[3].

Table 2.2: Labels used to retrieve information from KG triples.

Narrative dimension	SEM predicate	Labels
Who	`sem:hasActor`	person, combatant, commander, participant
When (begin)	`sem:hasBeginTimeStamp`	start time, date, point in time
When (end)	`sem:hasEndTimeStamp`	end time
Where	`sem:hasPlace`	place, location, country
Part of	`sem:subEventOf`	partof, part of
Part of (inverse)	`sem:hasSubEvent`	has part, significant event

ChronoGrapher goes beyond EventKG and adds structured information from text. We mainly use the frame semantics theory [14], which introduce frames, that can be seen as templates with roles that need to be filled in. Some event-centric KGs built from news articles [42, 28] also used frames. Although EventKG contains text events and links them to entities, it does not provide the finer-grained semantic structure of frames. One example of frames is the `Change_of_ leadership` frame[4]. It describes "the appointment of a `New_leader` or removal from office of an `Old_leader`". Semantic roles such as `Old_leader` and `New_leader` are called frame elements.

The step of the extraction of frames and frame elements is done automatically. We use a pre-trained transformer-based model [7] to extract the frames and the frame elements. We use the NIF [18] ontology to encode text content, and DBpedia Spotlight [17] to extract the entities from frame elements. Each frame and frame element is added to the knowledge

[3] The rules can be found in the code, in the file "src/build_ng/generic_kb_to_ng.py

[4] `https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Change_of_leadership`

graph. We lastly link these frames to Framester [16], and generate annotations using the same structure as Framester[5][6].
An example is shown in Figure 2.5. It begins with the abstract from `dbr:Coup_of_18_Brumaire`, which is broken down into sentences like `ex:S1`. The frame annotation `ex:FNAnnot1` is connected to the frame template `frame:Cause_to_end` and further linked to roles such as `ex:E1`, representing *the French Revolution*. This literal is then connected to the IRI entity `dbr:French_Revolution` via DBpedia Spotlight. The frame knowledge in the KG population (from RQ2) is used to answer the event-centric questions in RQ3.

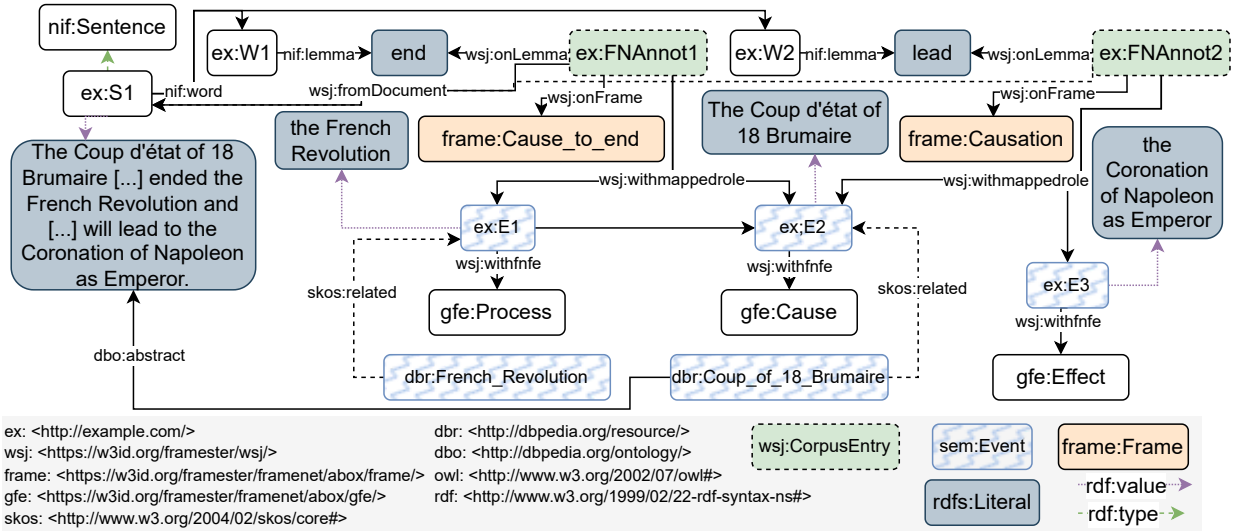

Figure 2.5: Example of frames extracted from text with the *Coup of 18 Brumaire*.

2.4 Evaluation

We present the setup (Section 2.4.1), and results on the quality of the method (Section 2.4.2) and the usefulness of the constructed event-centric KGs (Section 2.4.3). The quantitative evaluation (Section 2.4.2, RQ1 and RQ2) focuses on evaluating the quality of the method, both for the traversal (RQ1) and the KG population (RQ2). The evaluation for RQ1 is done automatically and can be reproduced. The evaluation for RQ2 is partly done automatically, and partly manually assessed. For RQ3, we conducted a preliminary user study to assess the potential of event-centric KGs on a QA setting, and we found encouraging qualitative results.

2.4.1 Experimental Setting (RQ1, RQ2)

Our experiments first aim to assess the quality of ChronoGrapher, both for relevant event extraction (RQ1) and for event-centric KG population (RQ2). For RQ1, we investigate the effects of filtering and ranking on the graph traversal approach. We first experiment under different configurations, with six distinct parameters for the search process: (1) scoring metric, (2) preference function (cfr. Section 2.3.2), (3-6) filters pertaining to WHAT, WHERE, WHEN, and WHO (Definition 2.3.2). The scoring metrics is one of the six options described above (Definition 2.3.2), whereas all the other parameters are boolean. We ran our experiments on two Ubuntu machines with 40 CPUs, and 348GiB and 320GiB of memory.

[5] `http://etna.istc.cnr.it/framesterpage/wsj/wsjpropnetannotations/CE_55094`
[6] The code for this part is in the "src/build_ng/frame_semantic.py" file.

Baselines. For RQ1 on relevant event extraction, we compare *chronographer* with three categories of system we implemented. We adapted LDSpider [33] and NautiLOD [18] to align with our method. The difference in performance between ChronoGrapher and the considered baselines is solely assessed on the event extraction part.

- *random-5*, *random-10* and *random-15*: at each iteration, n nodes are randomly selected to be expanded. We used $n \in \{5, 10, 15\}$. Parameters (1-6) are non applicable.
- *ldspider-m*: inspired by LDSpider [33], this search is a breadth-first search with some predefined predicates to be pruned out of the search space. Parameters (1) and (2) are not applicable, and (3-6) are set to 0.
- *nautilod-m*: inspired by NautiLOD [18], this baseline works as a breadth-first search combined with more complex filters than *ldspider-m*. Parameters (1) and (2) are not applicable, but we experiment with (3-6).

Evaluation. We used EventKG 3.1 [22] as ground truth since it was manually constructed. For RQ1 on relevant event extraction, we compare the sub-events extracted with our method to the ones accessible with EventKG. We use `SPARQL` queries to retrieve the start and end dates, and the sub-events of all events. For RQ2 on event-centric KG population, we construct a KG from EventKG that is directly comparable to our event-centric KGs. We compute F1, Precision and Recall as metrics.

Datasets. Our experiments focus on DBpedia (2021-09) [2], Wikidata (2021-03-05) [77] and YAGO4 [55] because EventKG was built on them. We use the HDT version of DBpedia, which we obtained from TriplyDB[7]. We obtain Wikidata from rdfhdt[8] and YAGO4 from their official website[9].

Table 2.3 shows information on events and their sub-events across datasets. All types of events are taken into account, which mainly include historical events, sports events and political events such as elections. We choose to retain solely the historical events with more than ten sub-events, as they offer more than a simple list of events. The final distribution of historical events in EventKG and their average number of outgoing links are presented in Table 2.4. Events with a higher number of sub-events tend to have a higher average number of features. The limited number of outgoing links for YAGO4 resulted in traversals that would end after one iteration. YAGO4 was built on Wikidata with an improved schema, without the full content from Wikidata, and we found that the ingoing links were mostly about creative works, and the outgoing links about metadata, discarded for the search. This resulted in a limited number of features. We thus only used Wikidata and DBpedia.

Table 2.3: Statistics on all events across datasets.

Dataset	Nb. of sub-events			
	= 1	>1	>10	Final
Wikidata	203,988	238,094	2,408	341 DBpedia
84,599	95,504	1,333	250	
YAGO4	70,738	76,682	993	306

[7] `https://triply.cc`
[8] `https://www.rdfhdt.org/datasets/`
[9] `https://yago-knowledge.org/downloads/yago-4`

Table 2.4: Number of filtered events and average number of features for events with more than n sub-events.

Dataset	Nb. of sub-events					Avg. nb. of features				
	>10	>20	>30	>50	>100	>10	>20	>30	>50	>100
Wikidata	341	208	147	91	47	237	273	290	323	381
DBpedia	275	158	106	66	37	122	133	141	147	160
YAGO4	306	180	126	76	44	5	4	5	5	5

Filters and preference function definition. The filters from Definition 2.3.2 were generic with some event-centric examples for the WHAT, WHERE, WHEN and WHO filters. Table 2.5 describes the predicates we used for each of the filters. Each filter relates to one of the core classes of the SEM ontology [75], making them event-centric. However, other filters can also be implemented by instantiating other variables, making the method easily expandable and usable for other tasks.

2.4.2 Quantitative Evaluation (RQ1, RQ2)

Parameter selection for the search (RQ1).

For ***chronographer*** and ***nautilod-m***, we first experimented on 12 events to select the most meaningful parameters to run the graph traversal on all events. Table 2.6 shows the 12 events used for parameter selection for the search. There were 192 possible combinations of parameters (6×2^5) for ***chronographer***, and 16 (2^4) for ***nautilod-m***.

We investigated on the correlation between our ranking and filtering parameters and the best F1 score. For ***chronographer***, we found a strong positive correlation between *domain_range* and higher F1 scores. This suggests that prioritizing events based on their domain range significantly improves event retrieval performance. We observed a weak but statistically significant correlation between *when* and the F1 score, suggesting that adding temporal information could help the search process. For ***nautilod-m***, we found a strong positive correlation between *what* and the best F1 score. The *what* filter indeed enables to remove more generic nodes from the search, which proves its efficiency here.
Table 2.7 shows the contributions of each filter individually. As an illustration, $what = 1$ means that only the experiments with this parameter activated, and all the others set to 0, are taken into account. For ***nautilod-m***, the parameter that yields the best result is *what*, with an average F1 of 0.36. The best average F1 score is obtained when all parameters are activated. On average, ***chronographer*** performs better or comparably to ***nautilod-m***, under all configurations. The parameter that yields the best results is *domain_range*.
For computational and performance reasons, we set the following parameters: **domain_range = when = what = where = who = 1**. We kept all metrics but the inverse ones for ***chronographer*** for the full experiments.

Lastly, Table 2.8 shows the results of all systems. We set a timeout of 10 hours for each experiment. Only 5 experiments from ***ldspider-m*** timed out. In these cases, we took the latest output of the search at that stage. Our systems achieve better result than the baselines. The search is furthermore more localised and efficient, since the output graphs and the runtime are smaller.

Search Results (RQ1).

Table 2.9 shows the final results on the remaining 604 events for all systems. The three metrics tend to improve when moving from the ***random*** baselines to our ***chronographer*** methods. ***random-5*** has the worst F1 scores, and ***chronographer-epof*** has the best ones. The differences are the most visible in the Recall scores, which means that the ***informed*** systems leave out less false negative than the random baselines. Precision scores for Wikidata tend to be higher than the ones for DBpedia, whereas it is the reverse for Recall. Therefore, the content retrieved in Wikidata is less noisy but also less complete. The ***random*** baselines and ***ldspider-m*** yield significantly bigger graphs than ***nautilod-m*** and the ***chronographer*** baselines, and take more time to run. ***nautilod-m*** performs better than the above baselines, however the size of the graphs and the runtime remain higher than the ***chronographer*** baselines (×4).

To summarise findings on relevant event extraction (RQ1), adding event-centric filters (WHAT, WHERE, WHEN, WHO) and a ranking step with an entropy predicate-object frequency heuristics helps to select relevant events while being more efficient time-wise and while exploring smaller regions of the graphs, as shown by the better performance of the ***chronographer*** systems in Table 2.9.

Event-centric KG Population Results (RQ2).

To generate our event-centric KGs, we use information from triples (i) and text (ii).

(i) Constructing event-centric KG from triples. We generate event-centric KGs for each event using all sub-events from the ground truth. We compare the number of common triples against EventKG. The results are shown in Table 2.10. ChronoGrapher achieves an overall F1 score of 67.2% and 76% for DBpedia and Wikidata. Precision scores for DBpedia are higher than for Wikidata, which means that the event-centric KGs are less noisy but less complete. For Wikidata, it is the opposite.

Table 2.11 shows the F1, Precision and Recall scores of the narrative graphs generated from the output of the search algorithm presented in Section 2.3.2. Our end-to-end system achieves an overall (for all predicates) F1 score of 51.7% and 49.2% respectively. As in Table 2.10, precision is higher for DBpedia, while recall tends to be higher for Wikidata. The results are furthermore lower than those in Table 2.10, which is expected since the output of the search also contains events that are not in the ground truth events. Consequently, for each of such event, all generated triples will not be in the ground truth from EventKG.

(ii) Constructing event-centric KG from text. We generate event-centric KGs from all the DBpedia events and sub-events that contain an abstract. The content that is extracted in this part is not present in EventKG. In total, this sums up to 9,438 events. On average, each event had 18 distinct frames and 48 instantiations of frames with at least one mapped role. The frames that appear the most in terms of number are: `Military` (26,681), `Hostile_encounter` (26,111), `Political_locales` (17,513), `Attack` (16,155), and `Leadership` (14,002). We are interested in events and relations between events, hence we focus on "causation" frames for a qualitative analysis, but many other frames are extracted. We manually annotate (1) whether there is a causation frame (2) whether the cause and the effect are correctly identified (3) whether the DBpedia entity is correct for 100 frame annotations. Out of 100 frames, 97 had at least one identified Cause or Effect. 85 were correctly identified as Causation frames. False positive causation frames often contain causative lemmas such as the noun "cause" that might have induced the model in error. Among the 171 retrieved frame elements, 151 (88.3%) were

correct. The majority of wrong frame elements resulted from false positive causation frames. Among the 94 DBpedia entities that were identified, 80 (85.1%) were correct.

To summarise findings on event-centric KG population (RQ2), we use both information extraction from triples and text. For triples, our system achieves F1 scores of 67.2% and 76% for DBpedia and Wikidata. For text, we conduct a qualitative evaluation on the "causation" frames and find that the system is able to retrieve information with good performance.

2.4.3 User Study (RQ3)

To evaluate the utility of event-centric KGs on a QA setting (RQ3), we designed 6 types of event-centric questions: summary of an event, causal explanations for an event, types of events within a time-range, sub-events of an event, events in which an actor participated, and events in which two actors both participated. We prompted GPT-4[10] in three ways: (a) `base`: base prompt, without context triples, (b) `db-kg`: `base` + context triples from DBpedia, and (c) `ec-kg`: `base` + context triples from the event-centric KG built by ChronoGrapher.
We gathered 10 participants for our user studies. We reached the participants of the user study throughout our research labs. The researchers who participated are based in Paris or Amsterdam, and they have a background in computer science and/or computational linguistics. They were not overly familiar with the questions we had them evaluated, and were familiar with LLMs systems and their pitfalls.
For the context triples, we manually implemented the `SPARQL` queries to retrieve triples from DBpedia or the event-centric KG[11]. The participants were asked to assess the quality of the answers with these criteria: (1) granularity of the info, (2) relevance to the question (3) succinctness and (4) diversity in content, on a scale of 1 to 5. We furthermore manually assessed (5) grounding to factual events. A QA system must most importantly give faithful content, making the groundedness metric (5) a priority. The relevance metric measures how well the answer provides a plausible answer to the question, but not its actual truthfulness. We designed 2 forms [12] [13] with 6 questions each, and with 3 answers per question corresponding to the three prompt types. We split the 10 participants evenly across these 2 forms. Each participant rated 18 answers (6 question types $\times$ 3 prompt types). Each prompt was assessed 60 times for metrics 1-4 (6×10), and 12 times for metric 5 (6×2). Given the scale, we solely focused on the French Revolution.
Table 2.12 presents the average scores for each metric and prompt type. The `base` prompt has the lowest groundedness which undermines its reliability, and and we found that it was prone to hallucinations, often about actors outside the LLM's implicit knowledge base. This might be linked to its lowest granularity and highest succinctness scores, due to having less content to work with. For the remainder of the analysis, we discard the `base` prompt because of its low groundedness score.
When comparing the triples-based prompts, `ec-kg` stands out with the highest groundedness (+27%) and succinctness (+8%) scores, indicating that its generated answers are both more faithful and concise. Additionally, `ec-kg` scores slightly higher in relevance (+2.5%) compared to `db-kg`, though `db-kg` achieves a higher diversity (+3.8%) score, likely due to its more varied context triples, and a slightly higher granularity score (+0.5%). Overall, `ec-kg` emerges as the most promising approach, balancing faithfulness, relevance, and conciseness effectively.

This first small user study thus hints that event-centric KGs can help an LLM provide more

[10] `https://openai.com/research/gpt-4`
[11] The queries can be found in the code: "experiments_run/usage_ng/sparql_queries.py"
[12] Form 1 link: `https://forms.gle/1SJTtTMUUM4uh7KT6`
[13] Form 2 link: `https://forms.gle/Rtgj7ucpdC1jm3tc6`

factual and concise answers to event-centric questions (2.85) than a base prompt (1.11) or a generic KG-based prompt (2.24), while maintaining succinctness.

2.5 Conclusion

We address the challenge of automatically constructing event-centric KGs from generic ones. We present ChronoGrapher that extracts relevant content and builds an event-centric KG. To extract relevant content (RQ1), ChronoGrapher contains an informed graph traversal strategy integrating event-centric filters to prune the search space and a ranking step for node prioritisation, yielding F1 scores of 0.78 and 0.68 for DBpedia and Wikidata. To generate event-centric KGs (RQ2), ChronoGrapher combines a rule-based system that achieves F1 scores of 0.67 and 0.76 for DBpedia and Wikidata, and information extraction from text. To evaluate the utility of event-centric KGs (RQ3), we conduct preliminary first experiments comparing different prompts on an event-centric QA setting, and show that prompts enriched with event-centric KG triples give more factual answers while maintaining succinctness (2.95 vs. 1.11 and 2.24). Future work will focus on improving ChronoGrapher in a number of directions, including expanding the search method to reach more sub-events using e.g. EventKG for additional type information; expanding the event-centric KG population beyond EventKG to tackle data noise and incompleteness; improving the modelling of complex entities; considering novel ontologies and vocabulary for the event representation (e.g. RDF-star); and use more sophisticated RAG-based methods for a larger user study. We also plan to work on additional quantitative and automatic metrics for a better evaluation of the event-centric KGs, since the long-term availability of the OpenAI service is uncertain and the human ratings are hard to reproduce.

Supplemental Material Statement: Source code for our system, the baselines we (re-)implemented and the experiments is available on Github[14]. The source code contains the predicate labels that were used to generate the event-centric KG (Section 2.3.3), pointers to download the datasets and more generic statistics on events and their sub-events (Section 2.4.1), the 12 events for parameter selection (Section 2.4.2), additional results on experiments on the end-to-end system combining the event extraction and the graph generation and the manual annotations for the causation frames (Section 2.4.2), and all code, prompts and data linked to the user study (Section 2.4.3).

[14] https://github.com/SonyCSLParis/graph_search_framework/tree/main

Table 2.5: Instantiated variables used for each of the filters and the preference function.

Name	Type	Dataset	Variable	Predicates
WHAT	predicate filter	DB	$\mathcal{R}'$	*rdf:type*
		WD		*wp:P31*
WHO	entity filter	DB	$\mathcal{E}'$	*dbo:Person*
			p	*rdf:type*
		WD	$\mathcal{E}'$	*wd:Q5*
			p	*wd:P31*
WHERE	entity filter	DB	$\mathcal{E}'$	*dbo:Place*
				dbo:Location
			p	*rdf:type*
		WD	$\mathcal{E}'$	*wd:P17*
				wp:P276
				wd:Q6256
			p	*wp:P31*
WHEN	temporal filter	DB	ps	*dbp:date*
				dbp:startDate
				dbp:birthDate
			pe	*dbp:endDate*
				dbp:deathDate
		WD	ps	*wp:P585*
				wp:P580
				wp:P569
			pe	*wp:P582*
				wp:P570
Preference Function	-	DB	d	*rdfs:domain*
			sc	*rdfs:subClassOf*
			t_e	*dbo:Event*
		WD	d	*wd:Q21503250*
			sc	*wp:P279*
			t_e	*wd:Q13418847*

Table 2.6: Events used for parameter selection.

World War I	American Indian Wars	Mediterranean and Middle East Theatre of World War II,
French Revolution	Cold War	European Theatre of World War II
Napoleonic Wars	Coalition Wars	European Theatre of World War I
Pacific War	Russian Civil War	Yugoslav Wars

Table 2.7: Mean F1 score for individual parameters. *no filter* and *all* means that all parameters are set to 0 and 1 respectively. *pf*, *pof*, *epf*, *epof*, *ipf* and *ipf* are the six possible scoring metrics (Section 2.4.1).

	nautilod-m	*chronographer*					
		pf	*pof*	*epf*	*epof*	*ipf*	*ipof*
domain_range = 1	N/A	0.66	**0.77**	0.66	**0.77**	0.66	0.65
what = 1	0.36	0.50	0.48	0.37	0.39	0.25	0.27
where = 1	0.28	0.47	0.48	0.33	0.39	0.25	0.23
when = 1	0.27	0.47	0.48	0.45	0.43	0.29	0.30
who = 1	0.25	0.43	0.50	0.39	0.48	0.26	0.26
no filter	0.31	0.46	0.48	0.34	0.39	0.24	0.24
all	0.53	0.69	**0.77**	0.69	**0.77**	0.69	0.72

Table 2.8: Results comparison of various systems.

System	F1	Precision	Recall	Nb_Nodes	Nb_Preds	Nb_Triples	Runtime
random-5	0.24	0.68	0.16	40,356	70	40,915	38'00"
random-10	0.31	0.54	0.24	1,554	70	**2,027**	43'00"
random-15	0.28	0.57	0.21	11,890	89	12,744	25'00"
ldspider-m	0.23	0.70	0.23	126,015	137	241,049	5'20"
nautilod-m	0.53	0.51	0.63	5,096	121	13,755	23'00"
chronographer-pf	0.69	0.70	0.71	874	21	2366	1'51"
chronographer-pof	**0.78**	**0.79**	0.77	750	**17**	2090	2'13"
chronographer-epf	0.69	0.70	0.72	857	20	2353	1'35"
chronographer-epof	**0.77**	0.78	**0.77**	**726**	**17**	2082	**1'42"**

Table 2.9: Systems results on the 604 events.*DB*: DBpedia, *WD*: Wikidata.

System	F1		Precision		Recall		Nb Nodes		Nb Preds		Nb Triples		Runtime	
	DB	WD	DB	WD	DB	WD	DB	WD	DB	WD	DB	WD	DB	WD
random-5	0.65	0.63	0.72	0.79	0.68	0.59	22,185	427	23	9	23,221	530	23'00"	11'00"
random-10	0.65	0.64	0.72	0.78	0.69	0.60	30,078	1,033	27	10	33,264	1,470	18'56"	11'33"
random-15	0.64	0.63	0.71	0.78	0.68	0.60	41,868	1,139	27	11	53,561	1,582	18'26"	16'43"
ldspider-m	0.64	0.63	0.73	0.78	0.68	0.61	131,607	4,992	52	20	239,548	7,389	27'05"	44'05"
nautilod-m	0.70	0.66	0.72	0.81	0.74	0.62	481	4,787	23	14	1,150	7,321	2'02"	39'26"
chronographer-pf	0.75	0.67	0.75	0.82	0.80	0.63	108	661	**5**	**6**	213	1,146	26"	9'38"
chronographer-pof	0.77	**0.68**	0.76	**0.83**	**0.82**	**0.64**	**97**	**612**	**5**	**6**	**196**	**863**	25"	8'43"
chronographer-epf	0.75	0.67	0.75	**0.83**	0.80	0.62	108	670	6	**6**	212	963	32"	9'07"
chronographer-epof	**0.78**	**0.68**	**0.77**	**0.83**	**0.82**	0.63	99	752	**5**	**6**	199	1,093	**24"**	**8'16"**

Table 2.10: Our event-centric KGs against EventKG. *DB*: DBpedia, *WD*: Wikidata.

Predicate	F1		Precision		Recall	
	DB	WD	DB	WD	DB	WD
`all`	67.2	76.0	92.1	72.1	52.9	80.4
`sem:hasActor`	65.0	43.6	90.4	58.1	50.7	34.9
`sem:hasBeginTimeStamp`	77.7	81.9	90.5	71.7	68.0	95.5
`sem:hasEndTimeStamp`	77.7	75.9	90.5	70.4	68.1	82.4
`sem:hasPlace`	63.6	90.3	95.0	92.1	47.8	88.6

Table 2.11: Metrics of generated narrative graphs compared to EventKG.

Pred	F1		Precision		Recall	
	DB	WD	DB	WD	DB	WD
`all`	51.7	49.2	79.3	41.1	38.4	61.4
`sem:hasActor`	48.5	27.6	76.5	35.1	35.5	22.8
`sem:hasBeginTimeStamp`	62.6	50.3	79.5	38.0	51.6	74.4
`sem:hasEndTimeStamp`	62.7	48.3	79.5	38.1	51.7	65.9
`sem:hasPlace`	48.2	59.2	81.6	50.7	34.2	71.2

Table 2.12: Average scores for metrics on the human evaluation. The Groundedness metric is the most important one as it assesses the faithfulness of the answers.

Type Prompt	Groundedness	Granularity	Relevance	Succinctness	Diversity
`base`	1.11 ± 1.47	(3.92 ± 0.96)	(4.18 ± 1.03)	(4.22 ± 0.92)	(3.83 ± 1.03)
`db-kg`	2.24 ± 1.6	**4.15** ± 0.9	4.02 ± 1.07	3.35 ± 1.01	**3.97** ± 1.09
`ec-kg`	**2.85** ± 1.86	4.13 ± 0.96	**4.12** ± 1.01	**3.62** ± 0.99	3.82 ± 1.03

Chapter 2: References

[AB1] Faisal Alkhateeb, Jean-François Baget, and Jérôme Euzenat. Extending sparql with regular expression patterns (for querying rdf). *Journal of web semantics*, 7(2):57–73, 2009.

[AB2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[AB3] Johanne Bakalara, Thomas Guyet, Olivier Dameron, André Happe, and Emmanuel Oger. An extension of chronicles temporal model with taxonomies-application to epidemiological studies. In *HEALTHINF 2021-14th International Conference on Health Informatics*, pages 1–10, 2021.

[AB4] Eva Blomqvist, Valentina Presutti, Enrico Daga, and Aldo Gangemi. Experimenting with extreme design. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 120–134. Springer, 2010.

[AB5] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. ICEWS Coded Event Data, 2015.

[AB6] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Valentina Presutti, and Chiara Veninata. Arco: The italian cultural heritage knowledge graph. In *International Semantic Web Conference*, pages 36–52. Springer, 2019.

[AB7] David Chanin. Open-source frame semantic parsing. *arXiv preprint arXiv:2303.12788*, 2023.

[AB8] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[AB9] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[AB10] Gong Cheng, Yanan Zhang, and Yuzhong Qu. Explass: exploring associations between entities via top-k ontological patterns and facets. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II 13*, pages 422–437. Springer, 2014.

[AB11] Laurens De Vocht, Sam Coppens, Ruben Verborgh, Miel Vander Sande, Erik Mannens,

and Rik Van de Walle. Discovering meaningful connections between resources in the web of data. In *LDOW*, 2013.

[AB12] Géraldine Del Mondo, Peng Peng, Jérôme Gensel, Christophe Claramunt, and Feng Lu. Leveraging spatio-temporal graphs and knowledge graphs: Perspectives in the field of maritime transportation. *ISPRS International Journal of Geo-Information*, 10(8):541, 2021.

[AB13] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. Rex: explaining relationships between entity pairs. *arXiv preprint arXiv:1111.7170*, 2011.

[AB14] Charles J Fillmore and Collin F Baker. Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*, volume 6, 2001.

[AB15] Erwin Filtz, María Navas-Loro, Cristiana Santos, Axel Polleres, and Sabrina Kirrane. Events matter: Extraction of events from court decisions. 2020.

[AB16] Valeria Fionda and Giuseppe Pirrò. Explaining graph navigational queries. In *The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I 14*, pages 19–34. Springer, 2017.

[AB17] Valeria Fionda, Giuseppe Pirrò, and Mariano P Consens. Querying knowledge graphs with extended property paths. *Semantic Web*, 10(6):1127–1168, 2019.

[AB18] Valeria Fionda, Giuseppe Pirrò, and Claudio Gutierrez. Nautilod: A formal language for the web of data graph. *ACM Transactions on the Web (TWEB)*, 9(1):1–43, 2015.

[AB19] Aldo Gangemi, Mehwish Alam, Luigi Asprino, Valentina Presutti, and Diego Reforgiato Recupero. Framester: A wide coverage linguistic linked data hub. In *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*, pages 239–254. Springer, 2016.

[AB20] Simon Gottschalk and Elena Demidova. Eventkg: A multilingual event-centric temporal knowledge graph. In *European Semantic Web Conference*, pages 272–287. Springer, 2018.

[AB21] Simon Gottschalk and Elena Demidova. Eventkg–the hub of event knowledge on the web–and biographical timeline generation. *Semantic Web*, 10(6):1039–1070, 2019.

[AB22] Simon Gottschalk and Elena Demidova. Eventkg, April 2020.

[AB23] Saiping Guan, Xueqi Cheng, Long Bai, Fujun Zhang, Zixuan Li, Yutao Zeng, Xiaolong Jin, and Jiafeng Guo. What is event knowledge graph: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[AB24] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training, February 2020. arXiv:2002.08909 [cs].

[AB25] Thomas Guyet. *Enhancing sequential pattern mining with time and reasoning.* PhD thesis, Université de Rennes 1, 2020.

[AB26] Steve Harris, Andy Seaborne, and E Prud'hommeaux. Sparql 1.1 query language. w3c recommendation (2013). *URL https://www. w3. org/TR/sparql11-query*, 2013.

[AB27] Olaf Hartig. Squin: a traversal based query execution system for the web of linked data.

In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1081–1084, 2013.

[AB28] Olaf Hartig and Jorge Pérez. Ldql: A query language for the web of linked data. *Journal of Web Semantics*, 41:9–29, 2016.

[AB29] Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating nlp using linked data. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, pages 98–113. Springer, 2013.

[AB30] José E Talavera Herrera, Marco Antonio Casanova, Bernardo Pereira Nunes, Giseli Rabello Lopes, and LAPP Leme. Dbpedia profiler tool: profiling the connectivity of entity pairs in dbpedia. In *Proceedings of the 5th International Workshop on Intelligent Exploration of Semantic Data (IESD 2016)*, pages 17–18. sn, 2016.

[AB31] José Eduardo Talavera Herrera. *On the connectivity of entity pairs in knowledge bases*. PhD thesis, Ph. D. Thesis, Department of Informatics, Pontifical Catholic University of …, 2017.

[AB32] Ioana Hulpuş, Narumol Prangnawarat, and Conor Hayes. Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I 14*, pages 442–457. Springer, 2015.

[AB33] Robert Isele, Jürgen Umbrich, Christian Bizer, and Andreas Harth. Ldspider: An open-source crawling framework for the web of linked data. In *Proceedings of the 2010 International Conference on Posters & Demonstrations Track*, volume 658, pages 29–32. Citeseer, 2010.

[AB34] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.

[AB35] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 792–802, 2021.

[AB36] Javier Guillot Jiménez, Luiz André P Paes Leme, and Marco A Casanova. Coepinkb: Evaluating path search strategies in knowledge bases. *Journal of the Brazilian Computer Society*, 28(1):13–25, 2022.

[AB37] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[AB38] Takahiro Kawamura, Shusaku Egami, Koutarou Tamura, Yasunori Hokazono, Takanori Ugai, Yusuke Koyanagi, Fumihito Nishino, Seiji Okajima, Katsuhiko Murakami, Kunihiko Takamatsu, et al. Report on the first knowledge graph reasoning challenge 2018. In *Joint International Semantic Technology Conference*, pages 18–34. Springer, 2019.

[AB39] Krys J Kochut and Maciej Janik. Sparqler: Extended sparql for semantic association

discovery. In *The Semantic Web: Research and Applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings 4*, pages 145–159. Springer, 2007.

[AB40] Hermann Kroll, Denis Nagel, and Wolf-Tilo Balke. Modeling narrative structures in logical overlays on top of knowledge repositories. In *International Conference on Conceptual Modeling*, pages 250–260. Springer, 2020.

[AB41] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics.

[AB42] Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer, 2013.

[AB43] 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. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, April 2021. arXiv:2005.11401 [cs].

[AB44] Zixuan Li, Xiaolong Jin, Saiping Guan, Wei Li, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. *arXiv preprint arXiv:2106.00327*, 2021.

[AB45] Pasquale Lisena, Daniel Schwabe, Marieke van Erp, Raphaël Troncy, William Tullett, Inger Leemans, Lizzie Marx, and Sofia Colette Ehrich. Capturing the semantics of smell: The odeuropa data model for olfactory heritage information. In *European Semantic Web Conference*, pages 387–405. Springer, 2022.

[AB46] Qingxia Liu, Gong Cheng, Kalpa Gunaratna, and Yuzhong Qu. Entity summarization: State of the art and future challenges. *Journal of Web Semantics*, 69:100647, 2021.

[AB47] Xuli Liu, Jihao Jin, Qi Wang, Tong Ruan, Yangming Zhou, Daqi Gao, and Yichao Yin. Patienteg dataset: bringing event graph model with temporal relations to electronic medical records. *arXiv preprint arXiv:1812.09905*, 2018.

[AB48] Steffen Lohmann, Philipp Heim, Timo Stegemann, and Jürgen Ziegler. The relfinder user interface: Interactive exploration of relationships between objects of interest. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 421–422, 2010.

[AB49] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[AB50] A. Merono, A. Ashkpour, M.G.J. van Erp, K. Mandemakers, L. Breure, A. Scharnhorst, K.S. Schlobach, and F.A.H. van Harmelen. Semantic technologies for historical research: A survey. *Semantic Web*, 6(6):539–564, 2015.

[AB51] Albert Meroño-Peñuela and Rinke Hoekstra. What is linked historical data? In *EKAW*, 2014.

[AB52] Joshua L Moore, Florian Steinke, and Volker Tresp. A novel metric for information retrieval in semantic networks. In *The Semantic Web: ESWC 2011 Workshops: ESWC*

2011 Workshops, Heraklion, Greece, May 29-30, 2011, Revised Selected Papers 8, pages 65–79. Springer, 2012.

[AB53] Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. A dataset for open event extraction in english. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1939–1943, 2016.

[AB54] Julia Noordegraaf, Marieke van Erp, Richard Zijdeman, Mark Raat, Thunnis van Oort, Ivo Zandhuis, Thomas Vermaut, Hans Mol, Nicoline van der Sijs, Kristel Doreleijers, Vincent Baptist, Charlotte Vrielink, Brenda Assendelft, Claartje Rasterhoff, and Ivan Kisjes. Semantic deep mapping in the amsterdam time machine: Viewing late 19th- and early 20th-century theatre and cinema culture through the lens of language use and socio-economic status. In Florian Niebling, Sander Münster, and Heike Messemer, editors, *Research and Education in Urban History in the Age of Digital Libraries - Second International Workshop, UHDL 2019, Dresden, Germany, October 10-11, 2019, Revised Selected Papers*, volume 1501 of *Communications in Computer and Information Science*, pages 191–212. Springer, 2019.

[AB55] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. Yago 4: A reasonable knowledge base. In *European Semantic Web Conference*, pages 583–596. Springer, 2020.

[AB56] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. nsparql: A navigational language for rdf. *Journal of Web Semantics*, 8(4):255–270, 2010.

[AB57] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. How Context Affects Language Models' Factual Predictions. February 2020.

[AB58] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language Models as Knowledge Bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[AB59] Giuseppe Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I 14*, pages 622–639. Springer, 2015.

[AB60] Juan L Reutter, Adrián Soto, and Domagoj Vrgoč. Recursion in sparql. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I 14*, pages 19–35. Springer, 2015.

[AB61] Mark O Riedl. Computational narrative intelligence: A human-centered goal for artificial intelligence. *arXiv preprint arXiv:1602.06484*, 2016.

[AB62] Adam Roberts, Colin Raffel, and Noam Shazeer. How Much Knowledge Can You Pack Into the Parameters of a Language Model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, November 2020. Association for Computational Linguistics.

[AB63] Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, Ger-

man Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37:132–151, 2016.

[AB64] Sebastian Schaffert, Thomas Kurz, Dietmar Glachs, Christoph Bauer, Fabian Dorschel, and Manuel Fernandez. The linked media framework. 2012.

[AB65] Jaspreet Singh, Wolfgang Nejdl, and Avishek Anand. History by diversity: Helping historians search news archives. In *Proceedings of the 2016 ACM on conference on human information interaction and retrieval*, pages 183–192, 2016.

[AB66] Laura Sinikallio, Senka Drobac, Minna Tamper, Rafael Leal, Mikko Koho, Jouni Tuominen, Matti La Mela, Eero Hyvönen, et al. Plenary debates of the parliament of finland as linked open data and in parla-clarin markup. In *3rd Conference on Language, Data and Knowledge, LDK 2021*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2021.

[AB67] Steven A Sloman. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3, 1996.

[AB68] Tarcísio Souza Costa, Simon Gottschalk, and Elena Demidova. Event-qa: A dataset for event-centric question answering over knowledge graphs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3157–3164, 2020.

[AB69] Steffen Stadtmüller, Sebastian Speiser, Andreas Harth, and Rudi Studer. Data-fu: A language and an interpreter for interaction with read/write linked data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1225–1236, 2013.

[AB70] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, March 2024. arXiv:2307.07697 [cs].

[AB71] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Learning to assess linked data relationships using genetic programming. In *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15*, pages 581–597. Springer, 2016.

[AB72] Francesca Tomasi. Artchives: a linked open data native catalogue of art historians' archives. 2021.

[AB73] Jürgen Umbrich, Aidan Hogan, Axel Polleres, and Stefan Decker. Improving the recall of live linked data querying through reasoning. *RR*, 7497:188–204, 2012.

[AB74] Chiel Van den Akker, Lora Aroyo, Agata Cybulska, Marieke Van Erp, Peter Gorgels, Laura Hollink, Cathy Jager, Susan Legene, Lourens van der Meij, Johan Oomen, et al. Historical event-based access to museum collections. *Applied Artificial Intelligence*, 25, 2010.

[AB75] Willem Robert Van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. Design and use of the simple event model (sem). *Journal of Web Semantics*, 9(2):128–136, 2011.

[AB76] Nikos Voskarides, Edgar Meij, Sabrina Sauer, and Maarten de Rijke. News article retrieval in context for event-centric narrative creation. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 103–112, 2021.

[AB77] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[AB78] Ian H Witten and David N Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. 2008.

[AB79] Jie Wu, Xinning Zhu, Chunhong Zhang, and Zheng Hu. Event-centric tourism knowledge graph—a case study of hainan. In *International conference on knowledge science, engineering and management*, pages 3–15. Springer, 2020.

[AB80] Harald Zauner, Benedikt Linse, Tim Furche, and François Bry. A rpl through rdf: Expressive navigation in rdf graphs. In *Web Reasoning and Rule Systems: Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings 4*, pages 251–257. Springer, 2010.