Neurosymbolic Methods for Dynamic Knowledge Graphs

Mehwish Alam, Genet Asefa Gesese, Pierre-Henri Paris

July 2024

Neurosymbolic Methods for Dynamic Knowledge Graphs

Mehwish ALAM [a,1] , Genet Asefa GESESE [b] and Pierre-Henri PARIS [c]
[a] *Télécom Paris, Institut Polytechnique de Paris, France*
[b] *FIZ Karlsruhe, Karlsruhe Institute of Technology, Germany*
[c] *University of Paris Saclay, France*

Abstract. Knowledge graphs (KGs) have recently been used for many tools and applications, making them rich resources in structured format. However, in the real world, KGs grow due to the additions of new knowledge in the form of entities and relations, making these KGs dynamic. This chapter formally defines several types of dynamic KGs and summarizes how these KGs can be represented. Additionally, many neurosymbolic methods have been proposed for learning representations over static KGs for several tasks such as KG completion and entity alignment. This chapter further focuses on neurosymbolic methods for dynamic KGs with or without temporal information. More specifically, it provides an insight into neurosymbolic methods for dynamic (temporal or non-temporal) KG completion and entity alignment tasks. It further discusses the challenges of current approaches and provides some future directions.

Keywords. Dynamic Knowledge Graphs, Temporal Knowledge Graphs, Dynamic Entity Alignment, Dynamic Knowledge Graph Completion

1. Introduction

Knowledge Graphs (KGs) [1] have gained attention in the past few years for representing information in a structured way, i.e., entities and relations. It has been used for various applications and domains from search engines and recommendation systems [2] to bio-informatics [3] and social sciences [4]. They capture relationships between entities in a way that is both human-readable and machine-processable, enabling advanced reasoning and inference capabilities. The significance of KGs lies in their ability to integrate vast amounts of heterogeneous data, providing a unified framework that supports comprehensive querying and analysis.

However, the real world is dynamic, with changes happening continuously—new entities emerge, existing relationships evolve, and facts that were once true may become outdated. In such scenarios, traditional static KGs fall short as they cannot accommodate these changes in real-time. This limitation has led to the development of *Dynamic Knowledge Graphs (DKGs)*, which can evolve by incorporating temporal information. Dynamic KGs update the data and track changes, enabling temporal queries and historical analysis.

[1] Corresponding Author: mehwish.alam@telecom-paris.fr.

The inclusion of temporality in KGs poses significant challenges. Firstly, the representation of time-sensitive data requires precise modeling to capture the validity period of facts. For example, without temporal information, a query asking about the president of the United States might return multiple results, leading to ambiguity. Temporal dynamics are crucial in contexts like legal documents, medical records, and financial transactions, where the timing of events plays a pivotal role in decision-making. Thus, preserving temporality in KGs is beneficial and necessary for ensuring the accuracy and relevance of the information.

Furthermore, learning representations over DKGs is essential for several reasons. It enables the development of models that can predict future changes, discover hidden patterns, and provide insights based on the temporal evolution of data. This is particularly valuable in predictive analytics, trend analysis, and anomaly detection, where understanding the temporal context can significantly enhance the accuracy of the results.

Entity alignment, the task of identifying equivalent entities across different datasets or KGs, becomes increasingly complex in dynamic settings. As entities and their attributes change over time, ensuring they are correctly matched requires sophisticated techniques that can handle temporal variations. This is crucial for maintaining data integrity, improving the accuracy of merged datasets, and enhancing the overall quality of the KG.

In summary, the dynamic and temporal dimensions of KGs are critical for accurately reflecting the complexity of real-world data. Addressing the challenges associated with these aspects can significantly advance the field of KG research and its applications, paving the way for more intelligent and responsive information systems.

This book chapter is structured as follows: Section 2 reviews related work and positions this chapter within the broader context. Section 3 provides fundamental definitions of the DKGs. Section 4 details how the dynamic information can be represented in the form of a KG. Following the discussion on representation techniques, the book chapter then moves on to explaining how neurosymbolic methods have been utilized for learning representations over DKGs for KG Completion (Section 5) and dynamic entity alignment (Section 6).

2. Related Work

One of the very recent studies provides an overview of the DKGs [5]. However, this chapter mostly focuses on providing a formal definition of different categories of DKGs along with neurosymbolic methods for DKGs, such as learning representations over such symbolic representations using methods based on neural networks along with the downstream tasks such as KG completion and entity alignment.

Several studies have been conducted on learning representations over KGs. For instance, various techniques for refining KGs are summarized in [6], including methods for KG completion. A categorization of static KG completion algorithms is given in [7]. In contrast, an exhaustive survey on multimodal KG embedding algorithms that utilize literal (text, numeric, image) information within the static KGs are presented in [8] along with an experimental comparison of these algorithms. Furthermore, a recent article [9] provides an overview of various KG completion tasks, such as transductive and inductive link prediction. It details methods incorporating background information from large

language models and discusses embeddings considering description logic axioms. However, all these studies are limited to static KGs and do not consider the dynamic aspects in a KG. In contrast, [10] surveys the methods for Temporal Knowledge Graph (TKG) completion. This book chapter builds on this by formally defining the fundamental differences between DKGs and TKGs. It provides a comprehensive overview of representation learning for these types of KGs, highlighting their unique characteristics and challenges. Additionally, the chapter discusses various downstream tasks associated with these graphs, such as KG completion and entity alignment, offering insights into the methodologies and applications tailored to DKGs and TKGs.

3. Preliminaries

This section formally defines static, temporal, and dynamic KGs with examples.

Definition 1 (Static Knowledge Graph) *Let $G = (E, R, L, F)$ be a directed labeled graph, where E, R, and L are the sets of entities, relations, and literals, respectively. $F \subseteq E \times R \times (E \cup L)$ represents a set of facts such that $f = (h, r, t)$ or $f = (h, r, l)$ represent one triple where $f \in F$, $h, t \in E$, $r \in R$, and $l \in L$.*

Example 1 *Consider a simple knowledge graph $G = (E, R, L, F)$ where:*

- ***Entities (E)**: $\{Barack\_Obama, USA\}$*
- ***Relations (R)**: $\{president\_of\}$*
- ***Literals (L)**: $\emptyset$*
- ***Facts (F)**: $\{(Barack\_Obama, president\_of, USA)\}$*

The example shows that Barack Obama is the president of the USA. Now, suppose we update the graph with the following:

- $E = E \cup \{Donald\_Trump\}$
- $R = R$
- $L = \emptyset$
- $F = F \cup (Donald\_Trump, president\_of, USA)$

In this static representation, the knowledge graph indicates that both Barack Obama and Donald Trump have been (or are) presidents of the USA. However, without temporal information, it is unclear when each individual held office.

The definition of a static KG can further be extended to a temporal KG, where each triple can have a time interval representing its temporal validity:

Definition 2 (Temporal Knowledge Graph) *Let $G = (E, R, L, T, Q)$ be a directed labeled graph, where E, R, L, and T represent the set of entities, relations, literals, and timestamps, respectively. $Q \subseteq E \times R \times (E \cup L) \times (T \cup \{\emptyset\})$ represents a set of facts such that $q = (h, r, t, [\tau_s, \tau_e])$ or $q = (h, r, l, [\tau_s, \tau_e])$ represent one quadruple where $q \in Q$, $h, t \in E$, $r \in R$, $l \in L$, and $\tau_s, \tau_e \in T \cup \{\emptyset\}$ and τ_s and τ_e represent the start and end time, i.e., $\tau_s \leq \tau_e$. If $\tau_s = \tau_e$ then it represents a point in time $\tau \in T$.*

Example 2 *Consider a simple knowledge graph $G = (E, R, L, F)$ where:*

- ***Entities (E)***: $\{$*Barack_Obama, USA*$\}$
- ***Relations (R)***: $\{$*president_of*$\}$
- ***Literals (L)***: $\{$*2009, 2017*$\}$ *(representing years)*
- ***Quadruples (Q)***: $\{$*(Barack_Obama, president_of, USA, [2009, 2017])*$\}$

The example shows when the fact that Barack Obama held office is a valid fact, i.e., from 2009 to 2017.

Definition 3 (Dynamic Knowledge Graph) *A dynamic knowledge graph G is a sequence of knowledge graphs $\{G_{t_0}, G_{t_1}, \ldots, G_{t_n}\}$ indexed by time steps $t_0, t_1, \ldots, t_n$ such that $t_0 < t_1 < \cdots < t_n$ and $\forall t_i, t_i \in T$, where T is the set of all timestamps. Each graph G_{t_i}, at time step t_i, can be either a static knowledge graph (see Definition 1) or a temporal knowledge graph (see Definition 2).*

The graph G_{t_i} represents a snapshot of the dynamic graph G at time t_i. The transition between two consecutive snapshots $G_{t_{i-1}}$ and G_{t_i} can be characterized by changes in entities, relations, literals, and facts. Formally:

$$E_{t_i} = E_{t_{i-1}} \cup E^{+}_{t_i} \setminus E^{-}_{t_{i-1}} \quad \textit{where } E^{+}_{t_i} \textit{ and } E^{-}_{t_{i-1}} \textit{ represent the additions and removals of entities},$$

$$R_{t_i} = R_{t_{i-1}} \cup R^{+}_{t_i} \setminus R^{-}_{t_{i-1}} \quad \textit{where } R^{+}_{t_i} \textit{ and } R^{-}_{t_{i-1}} \textit{ represent the additions and removals of relations},$$

$$L_{t_i} = L_{t_{i-1}} \cup L^{+}_{t_i} \setminus L^{-}_{t_{i-1}} \quad \textit{where } L^{+}_{t_i} \textit{ and } L^{-}_{t_{i-1}} \textit{ represent the additions and removals of literals},$$

$$G_{t_i} = G_{t_{i-1}} \cup G^{+}_{t_i} \setminus G^{-}_{t_{i-1}} \quad \textit{where } G^{+}_{t_i} \textit{ and } G^{-}_{t_{i-1}} \textit{ represent the additions and removals of facts}.$$

Additionally, if G_{t_i} is a temporal knowledge graph, it can include changes in the temporal intervals of the quadruples.

Example 3 *Consider the first snapshot of a simple knowledge graph $G_0 = (E_0, R_0, L_0, F_0)$ where:*

- ***Entities*** *(E_0)*: $\{$*Barack_Obama, USA*$\}$
- ***Relations*** *(R_0)*: $\{$*president_of*$\}$
- ***Literals*** *(L_0)*: $\emptyset$
- ***Facts*** *(F_0)*: $\{$*(Barack_Obama, president_of, USA)*$\}$

We can see that Barack Obama is the president of the USA as in the first example. Now, suppose we update the graph with the following new snapshot $G_1 = (E_1, R_1, L_1, F_1)$:

- ***Entities*** *(E_1)*: $E_0 \setminus \{Barack\_Obama\} \cup \{Donald\_Trump\}$
- ***Relations*** *(R_1)*: R_0
- ***Literals*** *(L_1)*: $L_0 = \emptyset$
- ***Facts*** *(F_1)*: $\{$*(Donald_Trump, president_of, USA)*$\}$

Now, Donald Trump is the president of the USA in the last snapshot G_1, which no longer contains information about Barack Obama.

Finally, consider the temporal KG of Example 2 as the first snapshot G_0 of a simple knowledge graph. It can be updated with the following new snapshot $G_1 = (E_1, R_1, L_1, F_1)$:

- ***Entities*** *(E_1)*: $E_0 \cup \{Donald\_Trump\}$
- ***Relations*** *(R_1)*: R_0

- ***Literals*** *(L_1): $L_0 \cup \{2021\}$*
- ***Quadruples*** *(Q_1): $Q_0 \cup \{(Donald\_Trump, president\_of, USA, [2017, 2021])\}$*

This example is more complete because it keeps both facts with their respective time validity.

4. Representing Dynamic and Temporal Knowledge Graphs

This section presents the different techniques to represent temporal information in knowledge graphs and that can be used for temporal KGs or dynamic KGs.

4.1. Techniques to Represent Temporal Information in Knowledge Graphs

The first nucleus to represent time is the use of various XML Schema Definition [11] (XSD) datatypes, such as `xsd:dateTime`, `xsd:duration`, `xsd:date`, and others, that allows for precise representation of temporal data, including specific moments, durations, calendar dates, and individual time components, in KGs. These datatypes have been designed to facilitate the accurate and standardized encoding of temporal information, enabling robust temporal querying and reasoning.

4.1.1. Temporal Properties

Temporal properties directly incorporate time into relationships within a KG. These properties provide inherent temporal aspects to the entities and relationships they describe. For example (please note that all examples are provided in the Turtle syntax[2]):

```
:Alice :birthDate "1990-01-01"^^xsd:date .
:Alice :employedAt :CompanyX .
:Alice :employmentStart "2020-01-01"^^xsd:date .
:Alice :employmentEnd "2022-01-01"^^xsd:date .
```

In this example, a simple temporal relation is represented using a time interval: Alice's employment relationship with CompanyX includes specific start and end dates, enabling queries about her employment duration. The advantage of this approach is its straightforwardness and direct annotation of temporal data within the relationships themselves. However, it may lack flexibility for more complex temporal scenarios because one need to know that "*employmentStart*" is related to "*employedAt*". Additionally, if one wants to keep track of Alice's employment history, this became more complex to represent. Temporal attributes can also be represented, e.g., we also know her birth date, which is a fixed point in time.

All cross-domain general-purpose KGs have some form of temporal information, e.g., Wikidata [12], DBpedia [13], YAGO [14,15], etc.

[2] `https://www.w3.org/TR/rdf12-turtle/`

4.1.2. Reification

Reification allows a triple to be treated as a subject of another triple, thereby attaching metadata such as temporal information. There are several ways to reify a triple.

The standard reification [16] is a built-in functionality in RDF, i.e., a resource is used to denote the fact and additional information about the statement can be added using the RDF vocabulary (the properties "*rdf:subject*", "*rdf:predicate*", and "*rdf:object*"). For instance:

```
_:statement rdf:type rdf:Statement .
_:statement rdf:subject :Alice .
_:statement rdf:predicate :knows .
_:statement rdf:object :Bob .
_:statement :since "2022-01-01"^^xsd:date .
```

Through reification, the relationship between Alice and Bob can be annotated with the date they became acquainted. This approach offers flexibility in adding metadata to existing triples but can lead to increased data redundancy and complexity.

Other possibilities to reify a fact are to use n-ary relations [17] or to use singleton properties [18]. The interested reader can refer to Hernández et al. [19] for more details and an in depth comparison of these approaches.

4.1.3. Time Ontology in OWL

The Time Ontology in OWL[3] provides a comprehensive framework for representing temporal concepts. This ontology includes classes and properties for describing instants, intervals, and durations. Thus, the ontology is designed to support complex temporal reasoning and querying following Allen's interval algebra [20]. For example:

```
@prefix ex: <http://example.org/ns#> .
@prefix time: <http://www.w3.org/2006/time#> .

# Define Alice as a person
ex:Alice a ex:Person ;
    ex:worksSince ex:JobStart .

# Define the job start time
ex:JobStart a time:Instant ;
    time:inXSDDateTime "2020-01-01T00:00:00Z"^^xsd:dateTime .

# Define a class for persons who have been working for more
# than 3 years
ex:LongTermEmployee a owl:Class ;
    owl:equivalentClass [
        owl:intersectionOf (
            ex:Person
            [
```

[3] https://www.w3.org/TR/owl-time/

```
                owl:onProperty ex:worksSince ;
                owl:someValuesFrom [
                    owl:restriction [
                        owl:onProperty time:before ;
                        owl:hasValue
                            "2021-01-01T00:00:00Z"^^xsd:dateTime
                    ]
                ]
            ]
        )
    ] .

# Reasoning
ex:Alice a ex:LongTermEmployee .
```

In this example, we use the OWL Time ontology to describe the start date of Alice's employment and perform basic reasoning to determine if she qualifies as a long-term employee.

The interesting definition is the class ex:LongTermEmployee used to represent individuals who have been working for more than three years. This class is specified using owl:equivalentClass to intersect ex:Person and a restriction on the ex:worksSince property. The restriction requires that the ex:worksSince property has a value indicating a work start date before January 1, 2021.

The reasoner determines that since Alice's work start date is "2020-01-01," which is before the cutoff date "2021-01-01," she meets the criteria for long-term employment.

4.1.4. Named Graphs and Quadruples

Named graphs [21] group triples into a single graph that can have metadata associated with it. This allows for temporal information to be added at the graph level. For example:

```
GRAPH <http://example.org/graph/2022-01-01> {
    :Alice :knows :Bob .
}
```

This structure facilitates temporal data management by associating entire sets of triples with specific temporal contexts. Named graphs are advantageous for segmenting data into manageable subgraphs but can introduce overhead in graph management and querying.

Quadruples[4] extend RDF triples by adding a fourth element, often used to represent the context or the named graph, which can include temporal information. Contrary to named graphs, quadruples are not a separate graph but a part of the RDF data model, thus, informations can be attached to the fact level instead of the whole graph. For example:

```
:Alice :knows :Bob "2022-01-01"^^xsd:date .
```

[4] https://www.w3.org/TR/rdf12-n-quads/

This quadruple includes a timestamp indicating when Alice and Bob's acquaintance began. The use of quadruples simplifies the representation of contextual information but requires a data store that supports quad storage and querying.

4.1.5. RDF-star

RDF-star[5] is an extension of the RDF data model that offers a more compact and intuitive way to represent complex statements, including those involving temporal information. Examples include:

```
<< :Alice :knows :Bob >> :since "2022-01-01"^^xsd:date .
<< :Alice :worksAt :CompanyX >> :startDate "2020-01-01"^^xsd:date ;
        :endDate "2022-01-01" .
<< :Alice :attended :ConferenceX >> :onDate "2023-06-15"^^xsd:date .
<< :Document123 :hasVersion :Version1 >> :timestamp
        "2024-01-01T10:00:00Z" .
<< :DataItem :createdBy :User123 >> :creationTime
        "2024-07-01T09:00:00Z" .
```

RDF-star enables a streamlined representation of temporal information, simplifying the construction and querying of knowledge graphs. Its compactness reduces data redundancy and complexity but requires support for RDF-star syntax and semantics in RDF processors.

4.1.6. Versioning

Maintaining historical data with timestamps is crucial for tracking changes over time. Versioning techniques ensure that the evolution of data is documented, enabling temporal queries and historical analysis. One can use simple approaches, such as adding a timestamp to each triple, or more complex methods, such as reification or named graphs. For example, using versioning:

```
:Document123 :hasVersion :Version1 .
:Version1 :timestamp "2024-01-01T10:00:00Z" .
```

More complex versioning systems have been proposed over time: Frommhold et al. [22] developed a Version Control System for KGs, emphasizing the need for efficient change detection across graphs, including handling blank nodes. Their system uses invertible patches to manage changes, supporting operations like revert and merge, and ensuring data integrity through secure hashing of patches. This approach is foundational, particularly in tracking changes in complex datasets and ensuring reliable data history.

Pelgrin et al. [23] provided a comprehensive survey and analysis of KG archiving systems, highlighting the lack of standardization and scalability in existing solutions. They introduced RDFev, a framework for studying the dynamicity of RDF data, which offers insights into dataset evolution at both low and high levels. Their work outlines the essential features of a fully-fledged RDF archiving system, emphasizing the need for robust support for concurrent updates, efficient query processing and comprehensive version control features like branching and tagging.

[5] https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

	Properties	Reification	Time Ontology	Named Graphs & Quadruples	RDF-star	Versioning
Temporal	X	X	X	X*	X	
Dynamic		X		X*	X	X

Table 1. Comparison of Temporal and Dynamic Capabilities Across Various RDF Techniques. The presence of a feature is indicated by an 'X'. The asterisk under Named Graphs & Quadruples denotes that a choice must be made between temporal and dynamic, depending on the semantics intended by the graph owner.

Table 1 provides a summary of the temporal and dynamic capabilities of the various RDF techniques presented in this section For instance, temporal properties give a straightforward way to annotate data with time, but they may not offer the flexibility needed for complex scenarios where relationships between different temporal attributes must be explicitly defined. Reification and Named Graphs/Quadruples allow for more detailed metadata, such as temporal context, versioning, or provenance, which can be crucial for historical analyses or data integrity. However, these methods can introduce data redundancy and increase the complexity of queries. Additionally, only one value can be used as a fourth value or as the name of the named graph.

The Time Ontology in OWL is particularly useful for those needing to model complex temporal relations, such as intervals and durations, and supports reasoning over these data types. This makes it ideal for applications requiring detailed temporal reasoning, such as historical data analysis or event tracking.

RDF-star offers a more compact and intuitive representation, which reduces data redundancy and simplifies the data structure. This can be particularly advantageous when dealing with large datasets or when the overhead of traditional reification methods becomes a concern. However, RDF processors must support RDF-star syntax and semantics, which might not be universally available.

Versioning techniques are essential for tracking changes over time, ensuring that a KG can evolve while maintaining a record of past states. This is particularly relevant in domains where data history is as important as the data itself, such as legal or historical research, or dynamic KG completion.

When choosing a method for representing temporal information in KGs, researchers and practitioners should consider the specific requirements of their application, including the complexity of the temporal relationships, the need for metadata, the size of the dataset, and the capabilities of the RDF processors they are using.

4.2. Prominent Knowledge Graphs

We now describe some of the most prominent general-purpose KGs and how they represent temporal information.

4.2.1. DBpedia

DBpedia [13] incorporates temporal aspects into its dataset through properties that denote specific time-related data. For example, temporal properties such as *dbo:birthDate* and *dbo:deathDate* indicate birth and death dates for people, while other properties capture periods of activity or events, such as *dbo:productionStartDate* and *dbo:productionEndDate* for manufacturing or production events. This inclusion of temporal data allows for queries about the duration of events or the temporal relationships

between entities. However, representing complex temporal scenarios might require additional context or metadata, as basic temporal properties might not inherently indicate their relation to other properties without such context.

DBpedia handles versioning primarily by maintaining historical data, which allows tracking changes over time. The DBpedia Live system plays a crucial role in this aspect, as it continuously synchronizes with Wikipedia to update the dataset with the latest information. This system utilizes a stream of updates from Wikipedia, enabling DBpedia to reflect recent edits with minimal delay, typically within a few minutes. Versioning also exists in DBpedia through snapshots, ensuring that historical changes are documented, which is essential for conducting temporal queries and historical analyses.

4.2.2. YAGO

The second iteration of YAGO [14] incorporates temporal dimensions into its KG by assigning existence times to entities and facts. This is achieved using specific relations such as *wasBornOnDate*, *diedOnDate*, *wasCreatedOnDate*, and *wasDestroyedOnDate*, which are standardized under generic entity-time relations like *startsExistingOnDate* and *endsExistingOnDate*. For events lasting a single day, YAGO2 uses *happenedOnDate*, which is a sub-property of both *startsExistingOnDate* and *endsExistingOnDate*. Facts are assigned a time point if they are instantaneous events or a time span if they have an extended duration with known beginning and end. This approach allows the system to deduce the temporal scope of entities and events, providing a comprehensive temporal annotation across the dataset.

For example, entities such as people are assigned birth and death dates, while artifacts and groups have creation and potential destruction dates. In cases where the data is incomplete or not applicable (e.g., abstract concepts or mythological entities), no temporal data is assigned, adhering to a conservative approach. This temporal framework is essential for enabling time-based queries and analyses, such as determining the lifespan of individuals or the duration of historical events.

YAGO 4.5 [15] builds upon the temporal framework of previous versions by enhancing its capacity to handle temporal information more flexibly and in more detail. The integration of temporal data into YAGO 4.5 utilizes the RDF-star model, allowing for more intricate annotations of facts. This temporal tagging is crucial for representing the evolving nature of the KG, allowing for dynamic queries that reflect changes over time. All YAGO versions are accessible only as archives, as the live version is not publicly available.

4.2.3. Wikidata

Wikidata [12] utilizes a system of qualifiers to provide additional context to statements, including temporal information. Qualifiers in Wikidata are versatile and can specify details such as the period during which a statement is valid. For instance, in the following example, Douglas Adams's education at St John's College is annotated with start and end dates, providing a temporal dimension to his academic history:

```
wd:Q42 wdt:P69 wd:Q3918 .
wd:Q42 p:P69 _:statement .
_:statement ps:P69 wd:Q3918 .
_:statement pq:P580 "1971-10-01T00:00:00Z"^^xsd:dateTime .
```

```
_:statement pq:P582 "1974-06-01T00:00:00Z"^^xsd:dateTime .
```

In this data snippet, '*wd:Q42*' represents Douglas Adams, '*wd:Q3918*' represents St John's College, '*p:P69*' and '*ps:P69*' denote the property for educational institutions attended, and '*pq:P580*' and '*pq:P582*' provide the start and end times of the education period, respectively.

Qualifiers in Wikidata allow for highly detailed annotations of statements, including dates, locations, and other contextual information. This feature supports complex data representation but requires sophisticated data parsing and querying mechanisms to retrieve and interpret the full context of the data.

Wikidata's approach to versioning involves maintaining a complete history of all edits, thus allowing for comprehensive tracking of changes over time. This version history is essential for verifying data provenance and understanding the evolution of knowledge within the graph. Unlike other KGs, Wikidata's version history is openly accessible, providing a transparent view of the data's editorial process.

Furthermore, Wikidata is a DKG, continuously updated by a large community of contributors. This live updating system ensures that Wikidata remains current, reflecting the latest information and corrections as they become available. The dynamic nature of Wikidata, combined with its detailed versioning and temporal annotations, supports both real-time applications and historical research, making it a versatile tool for a wide range of uses.

4.2.4. EventGraph

EventKG [24,25], a multilingual event-centric TKG, focuses heavily on capturing and representing temporal relations and events. The system models events using a canonical representation that incorporates both the start and end times of events, leveraging a variety of data sources, including Wikidata, DBpedia, and YAGO. Temporal information in EventKG is linked to entities and events, allowing for detailed historical analysis and event tracking. The model uses properties such as *sem:hasBeginTimeStamp* and *sem:hasEndTimeStamp* to define the temporal span of events and entities' involvement in those events.

Additionally, EventKG employs the Simple Event Model (SEM) as its foundational schema, which it extends to represent temporal relations better. This includes modeling not only event-entity relationships but also complex temporal relations between multiple entities or events, such as sequences of events and their hierarchical relationships. This comprehensive temporal modeling enables sophisticated queries and analyses concerning the temporal dynamics of historical and contemporary events.

Using named graphs and quadruples, EventKG also addresses versioning indirectly by maintaining a provenance framework that tracks the sources and version history of the data it integrates. Provenance information is critical in EventKG, as it helps verify the accuracy and credibility of the temporal data. Each piece of information, including temporal annotations, is associated with metadata that details its origin, the specific version of the source data, and the extraction date.

5. Dynamic Knowledge Graph Completion

In the existing literature [26], TKGC methods are generally classified into two categories: interpolation-based and extrapolation-based. Interpolation-based methods aim to predict missing knowledge by leveraging existing quadruplets. In contrast, extrapolation-based methods are designed for continuous TKGs, enabling prediction of future events by learning embeddings from previous or historical snapshots of entities and relations. This work treats interpolation methods as TKGC for TKGs, while extrapolation-based methods are applied to dynamic KGs, as defined in Section 3.

5.1. Temporal Knowledge Graph Completion (TKGC)

Temporal Knowledge Graphs (TKGs) often contain millions or billions of quadruplets, but they are typically incomplete for several reasons. First, extracting information from unstructured text sources can be error-prone, resulting in incomplete data. Second, capturing or integrating all available information, particularly from diverse or complex sources, is challenging. Third, the sources themselves may lack comprehensiveness or be biased, leading to selective inclusion of information and omission of other relevant facts. Lastly, the dynamic nature of information, with knowledge continuously evolving, contributes to these gaps.

The KGC task – commonly known as link prediction (LP) – aims to predict missing links by utilizing existing information. Various techniques have been developed for this task. However, a significant limitation of these methods is their difficulty in capturing the temporal dynamics of facts. They are typically designed for static knowledge graphs (KGs) that assume facts do not change over time. Therefore, these methods are ineffective when applied to TKGs, as they overlook the crucial temporal information inherent in TKGs. Temporal KGC (TKGC) methods have emerged to enhance LP accuracy by addressing the limitations of traditional KGC methods. TKGC methods improve upon these by incorporating timestamps of facts into the learning process in addition to the facts themselves. For instance, with TKGC, it is possible to infer the fact `(Gerhard Schröder, succeeded, Angela Merkel, 2005)` using the existing quadruplets `(Angela Merkel, Chancelor of, Germany, [2005 to 2021])` and `(Gerhard Schröder, Chancelor of, Germany, [1998 - 2005])`.

TKGC aims to predict possible links between two entities at a specific time. This can be accomplished in different ways: i) *tail prediction* - given the head and relation at a certain time, predicting the tail entity ($< h, r, ? >$), ii) *head prediction* - given the tail and relation at a particular time, predicting the head entity ($<?, r, t >$), iii) *relation prediction* - given the head and tail at a certain time, predicting the relation ($< h, ?, t >$). Inspired by the survey in [27], TKGC methods could be categorized into timestamps-dependent-based TKGC, timestamps-specific functions-based TKGC, and deep learning-based TKGC methods. A summary of these TKGC methods is provided in Table 2. Timestamps dependent-based TKGC methods such as TuckERTNT [28], TTransE [29], ST-TransE [30], T-SimplE [31], Canonical Polyadic (CP) [32], TKGFrame [33], TBDRI [34] associate timestamps to corresponding entities and relations to capture their evolution without directly manipulating the timestamps.

Timestamps-specific Functions-based TKGC methods use specialized functions, such as diachronic embedding, Gaussian, and transformation function, to learn em-

beddings for timestamps. Transformation functions-based TKGC methods include BoxTE [35], SPLIME [36], TARGCN [37], TASTER [38], Time-LowFER [39], Goel et al. [40], and DEGAT [41]. Complex embedding functions-based TKGC methods include ChronoR [42], TComplEx and TNTComplEx [43], TeRo [44], TGeomE [45], TeLM [46], RotateQVS [47], ST-NewDE [48], BiQCap [49],HA-TKGE [50], STKE [51], and HTKE [52]. Non-linear embedding functions-based TKGC methods iclude DyERNIE [53], ATiSE [54], HERCULES [55], and TKGC-AGP [56].

Deep learning-based TKGC methods capture the evolution of entities and relations by encoding timestamps using deep learning algorithms. These methods can be further grouped into i) Timestamps-Specific Space such as HyTE [57], HTKE [52], TRHyTE [58], BTHyTE [59], ToKEi [60], SANe [61], and QDN [62], ii) Long Short-Term Memory (LSTM)-based TKGC methods such as TA-TransE and TA-DistMult [63], TDG2E [64], Ma et al [65], CTRIEJ [66], and TeCre [67], and iii) Temporal constraint-based TKGC methods such as Chekol et al. [68], Kgedl [69], T-GAP [70], TempCaps [71], RoAN [72], and TAL-TKGC [73].

Training procedure: Given a TKG G and a set of facts Q in G, a scoring function $g(q)$ is defined to assign a factual score for a fact or a true quadruplet $q \in Q$. A negative sampling strategy [74] is employed to create a set of negative samples, i.e., factually incorrect quadruplets Q', to enhance the expressiveness of learned representations for the entities and relations in G.

Loss function: A loss function L aims at maximizing $g(q)$ for all $q \in Q$ and minimizing $g(q')$ for their negative samples $q' \in Q$. The following are the commonly used loss functions by TKGC approaches.

- **Margin-based ranking loss (MRL)** [75] enforces that the confidence in the corrupted quadruplet is lower than in the true quadruplet by a certain margin. MRL is computed as

$$L_{MRL} = \sum_{q \in Q} \left[\lambda + g(q) - \sum_{q' \in Q'} g(q')\right]_+, \tag{1}$$

  where $[x]_+ = max(x, 0)$ and $\lambda > 0$ is a margin hyperparameter.
- **Cross-entropy loss (CEL)** [76] also aims at obtaining a large gap between true quadruplets and the negative samples but without enforcing a fixed margin for all facts.

$$L_{CEL} = \sum_{q \in Q} \frac{exp(g(q))}{\sum_{q' \in Q'} exp(g(q'))} \tag{2}$$

- **Binary cross-entropy loss (BCEL)** [77] emphasizes the score of individual true quadruplets and negative samples as follows:

$$L_{BCEL} = \sum_{q \in Q \cup Q'} yg(q) + (1-y)g(q'), \tag{3}$$

  where $y = 1$ if $q \in Q$ and $y = 0$ otherwise.

Table 2. Comparative Analysis of the TKGC methods presented in Section 5.1. ILP, CS, and SKGC stand for Integer Linear Programming, Coordinate System, and Static Knowledge Graph Completion, respectively.

Technique	TKGC Methods	Remarks
Translation	TTransE [29], ST-TransE [30], BoxTE [35]	TTransE and ST-TransE struggle with handling temporally evolving facts. All are SKGC methods.
Tensor Decomp.	TuckERTNT [28], T-SimplE [31], Time-LowFER [39], QDN [62]	Time-LowFER ignores the rich contextual information in the graph structure. All except QDN are SKGC methods
ILP	TKGFrame [33]	TKGFrame can not handle complex scenarios like path queries.
Manifold	HERCULES [55], DyERNIE [53]	Both are SKGC methods. HERCULES does not show significant differences over benchmarks due to its limited use of temporal information.
Block Decomp.	TBDRI [34]	TBDRI is an SKGC method.
GCN	TARGCN [37]	TARGCN is an SKGC method.
GAT	DEGAT [41]	
Transformation	SPLIME [36]	SPLIME is an SKGC method.
Sparse Matrix	TASTER [38]	TASTER is an SKGC method.
Gaussian	ATiSE [54]	
Gaussian/Markov	TKGC-AGP [56]	
Polar CS	HA-TKGE [50], HTKE [52]	Both are SKGC methods.
Spherical CS	STKE [51]	
Complex Space	TNTComplEx [43], TeRo [44], ChronoR [42]	All are SKGC methods.
Quaternion Space	TeLM [46], RotateQVS [47], TGeomE [45]	All are SKGC methods. TeLM ignores the rich contextual information in the graph structure.
Dihedron Algebra	ST-NewDE [48]	ST-NewDE is an SKGC method.
Biquaternions/Manifold	BiQCap [49]	BiQCap is an SKGC method.
Time-Encoding	TA-DistMult [63], ToKEi [60]	Both are SKGC methods.
Time-Encoding/LSTM	TeCre [67]	TeCre is an SKGC method.
Temporal-Hyperp	HyTE [57], BTHyTE [59]	Both are SKGC methods.
Temporal-Hyperp/GRU	TRHyTE [58]	TRHyTE is an SKGC method.
Multi-semantic Space	SANe [61]	
GRU	TDG2E [64]	
GRU/Negative Sampling	CTRIEJ [66]	CTRIEJ is an SKGC method.
Path Reasoning	Kgedl [69]	
GNN	T-GAP [70]	
Capsule Network	TempCaps [71]	
Attention	RoAN [72], TALTKGC [73]	Both are SKGC methods.

Evaluation techniques: Typically, evaluating TKGC methods involves performing both head and tail predictions for each test quadruple q. For head prediction, the head entity in q is replaced with every possible entity in the TKG. Similarly, the tail entity is replaced with every possible entity for tail prediction. The scores generated by the scoring function for these modified quadruples are then ranked. Commonly used accuracy metrics for this evaluation include Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits@k.

Benchmark datasets: The commonly used benchmark datasets for TKGC can be categorized into those used for interpolation methods, such as ICEWS14 [63], ICEWS05-15 [63], GDELT [78], YAGO11k [57], YAGO15k [63], and Wikidata12k [57], and those used for extrapolation methods, including ICEWS14 [63], ICEWS18 [79], GDELT [78], WIKI [29], and YAGO [80].

5.2. Non-Temporal Dynamic KG Completion

Traditional methods for generating KG embeddings do not consider the evolving nature of a KG where new entities and relations are constantly being added to the KG. This requires training the embeddings over the new KG from scratch even if the changes are minimal which can lead to increased computational costs. Various attempts have been made to deal with this challenge. One of the first attempts to address this issue is online learning, which learns incrementally as new information arrives. However, one of the drawbacks of these methods of KG embedding is that they do not consider the problem of catastrophic forgetting. To mitigate this issue, continual or lifelong learning [81] methods were proposed, alleviating the problem of catastrophic forgetting in various tasks. In catastrophic forgetting, adaptation to a new distribution generally results in a largely reduced ability to capture the old ones.

Continual KG Embedding methods (CKGE) has recently received growing attention which perform fine-tuning with only new knowledge, leading to reduced training costs. These methods effectively alleviate the problem of catastrophic forgetting while learning the representations of the newly added knowledge. Continual Learning-based methods further target two kinds of solutions. First, the full-parameter fine-tuning paradigm which memorizes old knowledge by replaying a core old dataset or introducing additional regularization constraints. Although this paradigm effectively mitigates catastrophic forgetting, it significantly increases training costs, especially when handling large-scale KGs. Second, it adopts the incremental-parameter fine-tuning paradigm, with only a few parameters to learn emerging knowledge. This strategy may still lead to an increase in parameters and training time.

Recently, low-rank adapters such as LoRA have enabled efficient parameter fine-tuning and are used to reduce the training time in Large Language Models (LLMs). One of the very recent studies, FastKGE, introduces incremental low-rank adaptation mechanisms, IncLoRAs, to reduce training costs for continual learning for KG embeddings. The rest of the section discusses and compares these methods in detail.

5.2.1. Online Learning Based Methods

puTransE [82]. The current translational models (KG embedding models using translation-based scoring functions) for static KG embeddings have three significant

shortcomings. First, translational models are highly sensitive to hyperparameters such as margin and learning rate [83]. Second, for one triple, there is only one representation if the translation principle is followed, leading to low precision due to the congestion of entities and relations in vector space. Last but not least, new unseen entities or relations are not handled. One of the first methods to deal with the drawbacks mentioned above is Parallel Universe TransE (puTransE) – an extension of TransE – which learns multiple embeddings.

puTransE follows three steps: (i) triple selection (structurally and semantically aware triple selection), (ii) generation of random configurations, and (iii) learning embeddings.

Triple Selection Triple Selection includes semantically and structurally aware triple selection. To select semantically relevant entities, the puTransE samples a relation $r \in R$ and generates a set of entities $\{e_1, e_2, \ldots, e_n\} \in E$ of all entities containing r as either an outgoing or incoming edge. For structurally aware triple selection, the authors adopt the bidirectional random walk model using the entities selected as starting nodes. puTransE creates multiple embedding spaces each time new entities or relations arrive, and for each embedding, it defines a triple constraint both in count and diversity.

Generate Random Configurations Instead of defining a global configuration, puTransE generates different hyperparameter configurations for each embedding space. It randomly generates the value of the original TransE hyperparameters, i.e., margin and learning rate but also the values of triple constraint and number of training epochs.

Learning Embeddings puTransE uses a margin-based loss function.

puTransE, however, learns embeddings from the local parts of the KG, avoiding the retraining of the whole embedding space, which leads to the loss of global structural information in the learned embedding space. Additionally, since puTransE is an increment over TransE, it uses the scoring function of TransE, which does not work well for 1-to-N, N-to-1, and N-to-N relations.

DKGE [84] learns joint embedding for each entity and relation by considering their contextual information leading to an improvement over puTransE which only considers local information. The context of an entity e_1 is the one-hop neighborhood subgraph of the entity represented as $sg(e_1) \in E$. The context of a relation r_1 is represented by the relation path $p_1 = (r_1, r_2)$. The first step is to encode the contextual information of entity and relation as vector representations, which is then combined with the entity and relation embeddings (referred to as knowledge embeddings). After that, a scoring function and a loss function based on translation operations for parameter training are defined.

The algorithm uses Attentive GCN (AGCN) Model to effectively take into account the neighborhood information (modeled by GCN) and only use the information that may be important (modeled by the attention mechanism). Given x where $x \in EorR$ and its context, i.e., a subgraph with n vertices $\{v_i\}_{i=1}^{n}$, DKGE builds the adjacency matrix $A \in \mathbb{R}^{n \times n}$ and initializes the vertex feature matrix $H^{(0)} \in \mathbb{R}^{n \times d^0}$ where d^0 is the number of the initialized features for each vertex. Each row in $H^{(0)}$ is denoted as v_i. If x is an entity, then v_i is an entity and $\mathbf{v_i}$ is its contextual element embedding. When x is a relation, if v_i is a relation path consisting of two relations, then $\mathbf{v_i}$ is the sum of the contextual element embeddings of these two relations.

The final contextual subgraph embedding $\mathbf{sg}(x)$ of the subgraph $sg(x)$ is given by a weighted sum of the vectors of all vertices $\{v_i\}_{i=1}^{n}$ as follows:

$$\mathbf{sg}(x) = \sum_{i=n}^{n} \alpha_{i(x)} \mathbf{v_i} \tag{4}$$

The scoring function is given as follows:

$$f(h,r,t) = \| \mathbf{h^*} + \mathbf{r^*} - \mathbf{t^*} \|_{l_1} \tag{5}$$

where **h***, **r*** and **t*** are computed by Equation 4, and $\| \odot \|_{l_1}$ denotes the l_1 norm. A margin-based loss function is utilized for training. During online learning, at time step $T+1$, the knowledge embeddings and contextual element embeddings for new entities (relations) are randomly initialized according to a uniform distribution. For existing entities (relations), their knowledge embeddings and contextual element embeddings are carried over from the embedding results at time step T. These methods, however, do not consider literal information along with the structural information of a KG.

DKGC-JSTD [85] proposes a DKG completion model that jointly learns the structural and textual information of entities and relations based on a deep RNN. This model learns the embedding of an entity's name and parts of its text description to connect unseen entities to KGs. Deep memory network and association matching mechanism are used to extract semantic feature information and establish relevance between entity and relations from entity text-description. It then uses Recurrent Neural Networks to model the dependency between topology-structure and text description.

5.2.2. Continual Learning Based Methods

Various methods based on continual learning have been proposed. In this chapter, we discuss some of the fundamental algorithms.

Lifelong Knowledge Graph Embedding (LKGE) [86] learns the embedding of an entity or relation based on its masked first-order subgraph given as follows:

$$\bar{x}_i = MAE(\bigcup_{j=1}^{i} N_j(x)) \tag{6}$$

where $x \in E$ *or* R, and $N_j \subseteq D_j$ denotes the involved facts of x in the j-th snapshot and D represents the training set. $MAE()$ is an encoder that represents the input subgraph. The objective of the KG encoder is to align the entity or relation embedding with the reconstructed representation as follows:

$$L_{MAE} = \sum_{e \in E_i} \| e_i - \bar{e}_i \|_2^2 + \sum_{r} \in R_i \| r_i - \bar{r}_i \|_2^2 \tag{7}$$

Graph Convolution Networks (GCN) and Transformer are two common encoders. If they are updated for lifelong learning, the changed model parameters will affect the embedding generation of all old entities, including the involved old entities in new facts. This can lead to catastrophic forgetting of the previous snapshots. For this reason, the entity and relation embedding transition functions are used as encoders by LKGE, which do not introduce additional parameters and use TransE. TransE is leveraged to train the embeddings from the new data and update the knowledge transfer for each snapshot.

$$L_{new} = \sum_{(h,r,t)\in D_i} max(0, \gamma + f(h,r,t) - f(h',r,t')) \tag{8}$$

where γ is the margin. (h',r,t') is the embedding of a negative fact. The subject or object is randomly replaced with a random entity $e' \in E_i$ for each positive fact. The embeddings of unseen entities **e** and relations **r** are randomly initialized. To avoid catastrophic forgetting, regularization methods constrain the updates of the parameters. Accordingly, the loss function of regularization methods is given as:

$$L_{old} = \sum_{e\in E_{i-1}} w(e) \parallel e_i - e_{i-1} \parallel_2^2 + \sum_r \in R_{i-1} w(r) \parallel r_i - r_{i-1} \parallel_2^2 \tag{9}$$

where $w(x)$ is the regularization weight for x. The regularization weight is only computed once per snapshot. The overall learning objective is as follows:

$$L = L_{new} + \alpha L_{old} + \beta L_{MAE} \tag{10}$$

where α, β are hyperparameters for balancing the objectives.

One of the recent approaches uses low-rank adaptation for learning dynamic KG embeddings. Incremental Low-Rank Adaptation (IncLoRA) [87] operated in three stages. During the first graph layering stage, new entities and relations are divided into several layers based on the distance from the old graph and node degrees. The second stage involves IncLoRA learning, in which incremental LoRAs with adaptive rank allocation represent the embeddings of entities and relations in each layer. In the final stage, link prediction is performed, which composes all new LoRAs into a LoRA group and concat all LoRA groups and initial embeddings for inference.

Several previous studies have proposed using different mechanisms to learn the representations of the initial state of KGs and then incrementally learning the representations upon the arrival of new entities and relations. Some continual learning based algorithms use translational embeddings [88], progressive neural networks [89], incremental knowledge distillation [90], and graph attention networks [91].

6. Dynamic Entity Alignment

This section provides an overview of the temporal entity alignment methods between two KGs. In addition to TKG alignment, a few methods have been proposed considering new data while performing entity alignment.

6.1. Temporal Entity Alignment Methods

Most of the methods for temporal entity alignment use GCNs or GNNs to learn the representations of the entities alongside the temporal attention mechanism. In the following, we briefly overview the methods proposed so far.

In [92], the authors exploit both relation and temporal information for entity alignment by first creating a reverse link, i.e., introducing the inverse of the relations to handle the beginning and end of the relation. The method utilizes a time-aware attention mechanism with GCN to assign different weights to entities according to their relation and temporal information. On the other hand, TKG Entity Alignment via Representation Learning (Tem-EA) [93] incorporates temporal information with the help of Recurrent Neural Networks to learn temporal sequence representations. GCN and translation-based embedding models are used for learning representations over structural and attribute information for computing entity similarity separately on the two KGs to be aligned. These representations are combined using linear weighting. The concept of nearest-neighbor matching is performed to find the most similar entity pair based on the distance matrix.

A recent study [94] proposes a simple temporal information matching mechanism for entity alignment between TKGs. Most of the methods proposed prior to this use a time-aware attention mechanism to incorporate relational and temporal information into entity embeddings. This study assumes that learning the representations from the temporal information is unnecessary since a temporal matching mechanism, in addition to the GNN-based model, achieves better results. In [95], the authors perform entity alignment for TKGs via adaptive graph networks. More specifically, a time-aware graph attention network model is used as an encoder to aggregate the features and temporal relationships of neighboring nodes. DualMatch [96] is an unsupervised method that fuses the relational and temporal information for EA by encoding temporal and relational information into embeddings using a dual-encoder and combining both the information and transforming it into alignment using a novel graph-matching-based decoder called GM-Decoder.

6.2. Temporal and Evolving Entity Alignment Methods

Temporal Relational Entity Alignment (TREA) [97] learns alignment-oriented TKG embeddings and represents new emerging entities. The first step is to map entities, relations, and timestamps into an embedding space, and the initial feature of each entity is represented by fusing the embeddings of its connected relations and timestamps as well as its neighboring entities. A Graph Neural Network (GNN) is employed to capture intra-graph information, and a temporal relational attention mechanism is utilized to integrate relation and time features of links between nodes. Finally, a margin-based full multi-class log-loss is used for efficient training, and a sequential time regularizer is used to model unobserved timestamps.

Incremental Temporal Entity Alignment (ITEA) [98] targets the problem of temporality as well as evolving KGs. It uses a combination of knowledge distillation with the Graph Attention Network (GAT) and the GCN acting as the teacher and student models, respectively. The proposed model transfers knowledge from a complex model, the teacher, whose output (in terms of probabilities) is used to train a simpler model, the student. The teacher Model within knowledge distillation provides instructional guidance to the student model during its training phase.

As a first stage, entity embeddings are generated from the entity labels obtained using GloVe. For preserving structural information in the teacher model, a masked attention mechanism over the neighboring entities of the entity e_i. Time-aware structure embedding is obtained using a multi-head attention-based GNN. This allows the teacher model to not only learn the node features and time-aware structural information but also adjust their contributions adaptively based on the requirements of the specific task. The student model uses an importance-sampling strategy to select a small subset of nodes or neighbors in each layer, and the model is trained on mini-batches of nodes.

Entity alignment module aligns the newly incoming entities with the existing ones, and these alignments are achieved through string matching, structural similarity, etc. The margin-based logistic loss function measures the dissimilarity between the embeddings of aligned entities. The output from the teacher model is collected by passing the new data through the teacher model to compute its output probabilities, P_t. Then, the output of the student model is collected by passing the same data through the student model to compute its output probabilities, P_s. The total loss L can be written as:

$$L = \alpha_1 \times L_{standard} + \alpha_2 \times L_{distillation}$$

where α_1 and α_2 are weights that determine the relative importance of the standard loss and the distillation. Grid search is used to define possible values for α_1 and α_2. The standard loss compares the student's outputs P_s with the true labels y. $L_{standard}$ for a classification task is typically the cross-entropy loss given as:

$$L_{standard} = -\sum y \times log(P_s)$$

where the sum is over all classes, and y is a one-hot encoded vector of the true labels. The distillation loss that compares the student's outputs P_s with the teacher's outputs P_t. $L_{distillation}$ is the Kullback-Leibler divergence between the teacher's outputs and the student's outputs and is given as:

$$L_{distillation} = \sum P_t \times log(\frac{P_t}{P_s})$$

7. Discussion & Future Directions

This chapter formally defines various types of DKGs and explores how this knowledge can be represented within KGs. It then delves into different representation learning techniques for both non-temporal and temporal DKGs, focusing on tasks related to KG completion. Additionally, it covers methods for aligning temporal and dynamic KGs. The studies discussed in this chapter primarily rely on the triple structure or treat the KG as a graph by incorporating the contextual information of entities and relationships while largely overlooking schematic or ontological information [9]. Furthermore, the adaptability of these algorithms to domain-specific and real-world applications is limited, especially as KGs can expand to contain billions or even trillions of triples. Many of these

algorithms also fail to utilize the literal information within KGs. Only one of the recent studies has attempted to use Large Language Models (LLMs) as background information for TKGC [99] based on few-shot learning and concluded that LLMs do not bring significant increases in performance. However, the study misses the analysis of the results in the sense of how many predictions were hallucinations or overgenerations of LLMs for the specific task.

References

[1] Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, et al. Knowledge Graphs. ACM Comput Surv. 2022;54(4):71:1-71:37. Available from: `https://doi.org/10.1145/3447772`.

[2] Iana A, Alam M, Paulheim H. A survey on knowledge-aware news recommender systems. Semantic Web. 2024;15(1):21-82. Available from: `https://doi.org/10.3233/SW-222991`.

[3] Jonquet C, LePendu P, Falconer SM, Coulet A, Noy NF, Musen MA, et al. NCBO Resource Index: Ontology-based search and mining of biomedical resources. J Web Semant. 2011;9(3):316-24.

[4] Chen Y, Sack H, Alam M. Analyzing social media for measuring public attitudes toward controversies and their driving factors: a case study of migration. Soc Netw Anal Min. 2022;12(1):135. Available from: `https://doi.org/10.1007/s13278-022-00915-7`.

[5] Polleres A, Pernisch R, Bonifati A, Dell'Aglio D, Dobriy D, Dumbrava S, et al. How Does Knowledge Evolve in Open Knowledge Graphs? TGDK. 2023;1(1):11:1-11:59.

[6] Paulheim H. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web. 2017;8(3):489-508.

[7] Wang Q, Mao Z, Wang B, Guo L. Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Trans Knowl Data Eng. 2017;29(12):2724-43.

[8] Gesese GA, Biswas R, Alam M, Sack H. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? Semantic Web. 2021;12(4):617-47.

[9] Alam M, van Harmelen F, Acosta M. Towards Semantically Enriched Embeddings for Knowledge Graph Completion. CoRR. 2023;abs/2308.00081.

[10] Cai B, Xiang Y, Gao L, Zhang H, Li Y, Li J. Temporal Knowledge Graph Completion: A Survey. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China. ijcai.org; 2023. p. 6545-53.

[11] Thompson HS, Beech D, Maloney M, Mendelsohn N, et al. XML schema part 1: structures second edition. W3C recommendation. 2004;39.

[12] Vrandecic D. Wikidata: a new platform for collaborative data collection. In: Mille A, Gandon F, Misselis J, Rabinovich M, Staab S, editors. Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume). ACM; 2012. p. 1063-4.

[13] Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, et al. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web. 2015;6(2):167-95.

[14] Hoffart J, Suchanek FM, Berberich K, Weikum G. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artif Intell. 2013;194:28-61.

[15] Suchanek FM, Alam M, Bonald T, Chen L, Paris PH, Soria J. YAGO 4.5: A Large and Clean Knowledge Base with a Rich Taxonomy. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: Association for Computing Machinery; 2024. p. 131–140. Available from: `https://doi.org/10.1145/3626772.3657876`.

[16] McBride B. The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In: Staab S, Studer R, editors. Handbook on Ontologies. International Handbooks on Information Systems. Springer; 2004. p. 51-66.

[17] Erxleben F, Günther M, Krötzsch M, Mendez J, Vrandecic D. Introducing Wikidata to the Linked Data Web. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. vol. 8796 of Lecture Notes in Computer Science. Springer; 2014. p. 50-65.

[18] Nguyen V, Bodenreider O, Sheth AP. Don't like RDF reification?: making statements about statements using singleton property. In: Chung C, Broder AZ, Shim K, Suel T, editors. 23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014. ACM; 2014. p. 759-70.

[19] Hernández D, Hogan A, Krötzsch M. Reifying RDF: What Works Well With Wikidata? In: Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015. vol. 1457 of CEUR Workshop Proceedings. CEUR-WS.org; 2015. p. 32-47.

[20] Allen JF, Ferguson G. Actions and Events in Interval Temporal Logic. J Log Comput. 1994;4(5):531-79.

[21] Carroll JJ, Bizer C, Hayes PJ, Stickler P. Named graphs. J Web Semant. 2005;3(4):247-67.

[22] Frommhold M, Piris RN, Arndt N, Tramp S, Petersen N, Martin M. Towards Versioning of Arbitrary RDF Data. In: Proceedings of the 12th International Conference on Semantic Systems; 2016. .

[23] Pelgrin O, Galárraga L, Hose K. Towards fully-fledged archiving for RDF datasets. Semantic Web. 2021;12(6):903-25.

[24] Gottschalk S, Demidova E. EventKG: A Multilingual Event-Centric Temporal Knowledge Graph. In: Gangemi A, Navigli R, Vidal M, Hitzler P, Troncy R, Hollink L, et al., editors. The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings. vol. 10843 of Lecture Notes in Computer Science. Springer; 2018. p. 272-87.

[25] Gottschalk S, Demidova E. EventKG+TL: Creating Cross-Lingual Timelines from an Event-Centric Knowledge Graph. In: Gangemi A, Gentile AL, Nuzzolese AG, Rudolph S, Maleshkova M, Paulheim H, et al., editors. The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers. vol. 11155 of Lecture Notes in Computer Science. Springer; 2018. p. 164-9.

[26] Wang J, Wang B, Qiu M, Pan S, Xiong B, Liu H, et al. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. arXiv preprint arXiv:230802457. 2023.

[27] Cai B, Xiang Y, Gao L, Zhang H, Li Y, Li J. Temporal knowledge graph completion: A survey. arXiv preprint arXiv:220108236. 2022.

[28] Shao P, Zhang D, Yang G, Tao J, Che F, Liu T. Tucker decomposition-based temporal knowledge graph completion. Knowledge-Based Systems. 2022;238:107841.

[29] Leblay J, Chekol MW. Deriving validity time in knowledge graph. In: Companion proceedings of the the web conference 2018; 2018. p. 1771-6.

[30] Ni R, Ma Z, Yu K, Xu X. Specific Time Embedding for Temporal Knowledge Graph Completion. In: 2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC); 2020. p. 105-10.

[31] Lin L, She K. Tensor decomposition-based temporal knowledge graph embedding. In: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI); 2020. p. 969-75.

[32] He P, Zhou G, Zhang M, Wei J, Chen J. Improving temporal knowledge graph embedding using tensor factorization. Applied Intelligence. 2023;53(8):8746-60.

[33] Zhang J, Sheng Y, Wang Z, Shao J. TKGFrame: a two-phase framework for temporal-aware knowledge graph completion. In: Web and Big Data: 4th International Joint Conference, APWeb-WAIM 2020, Tianjin, China, September 18-20, 2020, Proceedings, Part I 4; 2020. p. 196-211.

[34] Yu M, Guo J, Yu J, Xu T, Zhao M, Liu H, et al. TBDRI: block decomposition based on relational interaction for temporal knowledge graph completion. Applied Intelligence. 2023;53(5):5072-84.

[35] Messner J, Abboud R, Ceylan II. Temporal knowledge graph completion using box embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36; 2022. p. 7779-87.

[36] Radstok W, Chekol M, Velegrakis Y. Leveraging static models for link prediction in temporal knowledge graphs. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE; 2021. p. 1034-41.

[37] Ding Z, Ma Y, He B, Tresp V. A simple but powerful graph encoder for temporal knowledge graph completion. arXiv 2022. arXiv preprint arXiv:211207791.

[38] Wang X, Lyu S, Wang X, Wu X, Chen H. Temporal knowledge graph embedding via sparse transfer matrix. Information Sciences. 2023;623:56-69.

[39] Dikeoulias I, Amin S, Neumann G. Temporal knowledge graph reasoning with low-rank and model-agnostic representations. arXiv preprint arXiv:220404783. 2022.

[40] Goel R, Kazemi SM, Brubaker M, Poupart P. Diachronic embedding for temporal knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34; 2020. p. 3988-95.

[41] Wang J, Zhu C, Zhu W. Dynamic embedding graph attention networks for temporal knowledge graph completion. In: International Conference on Knowledge Science, Engineering and Management. Springer; 2022. p. 722-34.

[42] Sadeghian A, Armandpour M, Colas A, Wang DZ. Chronor: Rotation based temporal knowledge graph

embedding. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35; 2021. p. 6471-9.
[43] Lacroix T, Obozinski G, Usunier N. Tensor decompositions for temporal knowledge base completion. arXiv preprint arXiv:200404926. 2020.
[44] Xu C, Nayyeri M, Alkhoury F, Yazdi HS, Lehmann J. TeRo: A time-aware knowledge graph embedding via temporal rotation. arXiv preprint arXiv:201001029. 2020.
[45] Xu C, Nayyeri M, Chen YY, Lehmann J. Geometric algebra based embeddings for static and temporal knowledge graph completion. IEEE Transactions on Knowledge and Data Engineering. 2022;35(5):4838-51.
[46] Xu C, Chen YY, Nayyeri M, Lehmann J. Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2021. p. 2569-78.
[47] Chen K, Wang Y, Li Y, Li A. Rotateqvs: Representing temporal information as rotations in quaternion vector space for temporal knowledge graph completion. arXiv preprint arXiv:220307993. 2022.
[48] Nayyeri M, Vahdati S, Khan MT, Alam MM, Wenige L, Behrend A, et al. Dihedron algebraic embeddings for spatio-temporal knowledge graph completion. In: European Semantic Web Conference. Springer; 2022. p. 253-69.
[49] Zhang S, Liang X, Li Z, Feng J, Zheng X, Wu B. Biqcap: A biquaternion and capsule network-based embedding model for temporal knowledge graph completion. In: International Conference on Database Systems for Advanced Applications. Springer; 2023. p. 673-88.
[50] Zhang J, Yu H. Hierarchy-Aware Temporal Knowledge Graph Embedding. In: 2022 IEEE International Conference on Knowledge Graph (ICKG). IEEE; 2022. p. 373-80.
[51] Wang S, Liu R, Shen L, Khattak AM. Stke: Temporal knowledge graph embedding in the spherical coordinate system. In: International Conference on Artificial Intelligence and Security. Springer; 2022. p. 292-305.
[52] He P, Zhou G, Liu H, Xia Y, Wang L. Hyperplane-based time-aware knowledge graph embedding for temporal knowledge graph completion. Journal of Intelligent & Fuzzy Systems. 2022;42(6):5457-69.
[53] Han Z, Ma Y, Chen P, Tresp V. Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. arXiv preprint arXiv:201103984. 2020.
[54] Xu C, Nayyeri M, Alkhoury F, Yazdi H, Lehmann J. Temporal knowledge graph completion based on time series gaussian embedding. In: The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I 19. Springer; 2020. p. 654-71.
[55] Montella S, Rojas-Barahona L, Heinecke J. Hyperbolic temporal knowledge graph embeddings with relational and time curvatures. arXiv preprint arXiv:210604311. 2021.
[56] Zhang L, Zhou D. Temporal knowledge graph completion with approximated Gaussian process embedding. In: Proceedings of the 29th International Conference on Computational Linguistics; 2022. p. 4697-706.
[57] Dasgupta SS, Ray SN, Talukdar P. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In: Proceedings of the 2018 conference on empirical methods in natural language processing; 2018. p. 2001-11.
[58] Yuan L, Li Z, Qu J, Zhang T, Liu A, Zhao L, et al. Trhyte: Temporal knowledge graph embedding based on temporal-relational hyperplanes. In: International Conference on Database Systems for Advanced Applications. Springer; 2022. p. 137-52.
[59] Liu K, Zhang Y. A Temporal Knowledge Graph Completion Method Based on Balanced Timestamp Distribution. arXiv preprint arXiv:210813024. 2021.
[60] Leblay J, Chekol MW, Liu X. Towards temporal knowledge graph embeddings with arbitrary time precision. In: proceedings of the 29th acm international conference on information & knowledge management; 2020. p. 685-94.
[61] Li Y, Zhang X, Zhang B, Ren H. Each snapshot to each space: Space adaptation for temporal knowledge graph completion. In: International Semantic Web Conference. Springer; 2022. p. 248-66.
[62] Wang J, Wang B, Gao J, Li X, Hu Y, Yin B. QDN: A quadruplet distributor network for temporal knowledge graph completion. IEEE Transactions on Neural Networks and Learning Systems. 2023.
[63] García-Durán A, Dumančić S, Niepert M. Learning sequence encoders for temporal knowledge graph completion. arXiv preprint arXiv:180903202. 2018.
[64] Tang X, Yuan R, Li Q, Wang T, Yang H, Cai Y, et al. Timespan-aware dynamic knowledge graph embedding by incorporating temporal evolution. IEEE Access. 2020;8:6849-60.

[65] Ma S, Li A, Zhao X, Song Y. Learning BiLSTM-based embeddings for relation prediction in temporal knowledge graph. In: Journal of Physics: Conference Series. vol. 1871. IOP Publishing; 2021. p. 012050.
[66] Li T, Wang W, Li X, Wang T, Zhou X, Huang M. Embedding uncertain temporal knowledge graphs. Mathematics. 2023;11(3):775.
[67] Ma J, Zhou C, Chen Y, Wang Y, Hu G, Qiao Y. Tecre: A novel temporal conflict resolution method based on temporal knowledge graph embedding. Information. 2023;14(3):155.
[68] Chekol M, Pirrò G, Schoenfisch J, Stuckenschmidt H. Marrying uncertainty and time in knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31; 2017. .
[69] Wang Y, Qiao Y, Ma J, Hu G, Zhang C, Sangaiah AK, et al. A novel time constraint-based approach for knowledge graph conflict resolution. Applied Sciences. 2019;9(20):4399.
[70] Jung J, Jung J, Kang U. Learning to walk across time for interpretable temporal knowledge graph completion. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining; 2021. p. 786-95.
[71] Fu G, Meng Z, Han Z, Ding Z, Ma Y, Schubert M, et al. TempCaps: a capsule network-based embedding model for temporal knowledge graph completion. In: Proceedings of the Sixth Workshop on Structured Prediction for NLP. Association for Computational Linguistics; 2022. p. 22-31.
[72] Bai L, Ma X, Meng X, Ren X, Ke Y. RoAN: A relation-oriented attention network for temporal knowledge graph completion. Engineering Applications of Artificial Intelligence. 2023;123:106308.
[73] Nie H, Zhao X, Yao X, Jiang Q, Bi X, Ma Y, et al. Temporal-structural importance weighted graph convolutional network for temporal knowledge graph completion. Future Generation Computer Systems. 2023;143:30-9.
[74] Yang Z, Ding M, Zhou C, Yang H, Zhou J, Tang J. Understanding negative sampling in graph representation learning. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining; 2020. p. 1666-76.
[75] Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. Advances in neural information processing systems. 2013;26.
[76] Li Z, Jin X, Guan S, Li W, Guo J, Wang Y, et al. Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. arXiv preprint arXiv:210600327. 2021.
[77] Liu Y, Hua W, Xin K, Zhou X. Context-aware temporal knowledge graph embedding. In: Web Information Systems Engineering–WISE 2019: 20th International Conference, Hong Kong, China, November 26–30, 2019, Proceedings 20; 2019. p. 583-98.
[78] Leetaru K, Schrodt PA. Gdelt: Global data on events, location, and tone, 1979–2012. In: ISA annual convention. vol. 2; 2013. p. 1-49.
[79] Jin W, Qu M, Jin X, Ren X. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. arXiv preprint arXiv:190405530. 2019.
[80] Mahdisoltani F, Biega J, Suchanek FM. Yago3: A knowledge base from multilingual wikipedias. In: CIDR; 2013. .
[81] Wang L, Zhang X, Su H, Zhu J. A Comprehensive Survey of Continual Learning: Theory, Method and Application. IEEE Trans Pattern Anal Mach Intell. 2024;46(8):5362-83.
[82] Tay Y, Luu AT, Hui SC. Non-Parametric Estimation of Multiple Embeddings for Link Prediction on Dynamic Knowledge Graphs. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. AAAI Press; 2017. p. 1243-9.
[83] Ali M, Berrendorf M, Hoyt CT, Vermue L, Galkin M, Sharifzadeh S, et al. Bringing Light Into the Dark: A Large-Scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework. IEEE Trans Pattern Anal Mach Intell. 2022;44(12):8825-45.
[84] Wu T, Khan A, Yong M, Qi G, Wang M. Efficiently embedding dynamic knowledge graphs. Knowl Based Syst. 2022;250:109124.
[85] Xie W, Wang S, Wei Y, Zhao Y, Fu X. Dynamic Knowledge Graph Completion with Jointly Structural and Textual Dependency. In: Qiu M, editor. Algorithms and Architectures for Parallel Processing - 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2-4, 2020, Proceedings, Part II. vol. 12453 of Lecture Notes in Computer Science. Springer; 2020. p. 432-48.
[86] Cui Y, Wang Y, Sun Z, Liu W, Jiang Y, Han K, et al. Lifelong Embedding Learning and Transfer for Growing Knowledge Graphs. In: Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023. AAAI Press; 2023. p. 4217-24.

[87] Liu J, Ke W, Wang P, Wang J, Gao J, Shang Z, et al.. Fast and Continual Knowledge Graph Embedding via Incremental LoRA; 2024. Available from: https://arxiv.org/abs/2407.05705.

[88] Song H, Park S. Enriching Translation-Based Knowledge Graph Embeddings Through Continual Learning. IEEE Access. 2018;6:60489-97.

[89] Daruna AA, Gupta M, Sridharan M, Chernova S. Continual Learning of Knowledge Graph Embeddings. IEEE Robotics Autom Lett. 2021;6(2):1128-35. Available from: https://doi.org/10.1109/LRA.2021.3056071.

[90] Liu J, Ke W, Wang P, Shang Z, Gao J, Li G, et al. Towards Continual Knowledge Graph Embedding via Incremental Distillation. In: Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada. AAAI Press; 2024. p. 8759-68.

[91] Kou X, Lin Y, Liu S, Li P, Zhou J, Zhang Y. Disentangle-based Continual Graph Representation Learning. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. Association for Computational Linguistics; 2020. p. 2961-72. Available from: https://doi.org/10.18653/v1/2020.emnlp-main.237.

[92] Xu C, Su F, Lehmann J. Time-aware Graph Neural Networks for Entity Alignment between Temporal Knowledge Graphs. CoRR. 2022;abs/2203.02150.

[93] Song X, Bai L, Liu R, Zhang H. Temporal Knowledge Graph Entity Alignment via Representation Learning. In: Database Systems for Advanced Applications - 27th International Conference, DASFAA 2022, Virtual Event, April 11-14, 2022, Proceedings, Part II. vol. 13246 of Lecture Notes in Computer Science. Springer; 2022. p. 391-406.

[94] Cai L, Mao X, Ma M, Yuan H, Zhu J, Lan M. A Simple Temporal Information Matching Mechanism for Entity Alignment between Temporal Knowledge Graphs. In: Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022. International Committee on Computational Linguistics; 2022. p. 2075-86.

[95] Li J, Song D, Wang H, Wu Z, Zhou C, Zhou Y. Entity alignment for temporal knowledge graphs via adaptive graph networks. Knowl Based Syst. 2023;274:110631.

[96] Liu X, Wu J, Li T, Chen L, Gao Y. Unsupervised Entity Alignment for Temporal Knowledge Graphs. In: Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023. ACM; 2023. p. 2528-38.

[97] Xu C, Su F, Xiong B, Lehmann J. Time-aware Entity Alignment using Temporal Relational Attention. In: Laforest F, Troncy R, Simperl E, Agarwal D, Gionis A, Herman I, et al., editors. WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022. ACM; 2022. p. 788-97.

[98] Li Y, Chen L, Liu C, Zhou R, Li J. Efficient and Effective Entity Alignment for Evolving Temporal Knowledge Graphs. In: Chen G, Khan L, Gao X, Qiu M, Pedrycz W, Wu X, editors. IEEE International Conference on Data Mining, ICDM 2023, Shanghai, China, December 1-4, 2023. IEEE; 2023. p. 349-58.

[99] Luo R, Gu T, Li H, Li J, Lin Z, Li J, et al. Chain of History: Learning and Forecasting with LLMs for Temporal Knowledge Graph Completion. CoRR. 2024;abs/2401.06072. Available from: https://doi.org/10.48550/arXiv.2401.06072.