TOMORROW BRINGS GREATER KNOWLEDGE: LARGE LANGUAGE MODELS JOIN DYNAMIC TEMPORAL KNOWLEDGE GRAPHS

Christian Di Maio
University of Pisa, Italy
christian.dimaio@phd.unipi.it

Andrea Zugarini
expert.ai, Italy
azugarini@expert.ai

Francesco Giannini
Scuola Normale Superiore, Italy
francesco.giannini@sns.it

Marco Maggini
University of Siena, Italy
marco.maggini@unisi.it

Stefano Melacci
University of Siena, Italy
stefano.melacci@unisi.it

ABSTRACT

Large Language Models (LLMs) have demonstrated exceptional capabilities in understanding and generating human-like text. In this paper, we leverage their powerful skills in the scope of lifelong learning agents. Instead of relying on fine-tuning procedures, we exploit Temporal Knowledge Graphs (TKGs) to continually store and update fresh information. In particular, we introduce a novel in-context learning approach called Continual In-context Knowledge LLM (CIK-LLM) capable of bridging an LLM with a dynamically changing TKG. The graph is updated whenever new knowledge becomes available, while the LLM is instructed to find the relational paths that are most relevant to the input instruction, with the goal of identifying smaller subgraphs of evidences. We propose to encode the subgraphs in a compressed-and-prompt-friendly manner, efficiently bridging the LLMs and TKGs. Then, the LLM provides an answer which is conditioned to the knowledge in the graph, exploiting its skills to support the reasoning process. We evaluate our approach on a TKG Question Answering benchmark which includes questions about events that happened at different times. The same questions are asked to models equipped with obsolete or incomplete information and models including progressively more up-to-date knowledge. CIK-LLM overcomes pre-trained LLMs, being able to immediately adapt to newly accumulated knowledge, and it reaches performances that are not far from the ones of a state-of-the art model which is trained not only exploiting LLMs but also large datasets of questions and answers. Furthermore, our model represents a valuable "forgetting-free" approach to quickly adapt an LLM to novel domains without any fine-tuning, QA datasets, or incremental learning procedures.

1 INTRODUCTION

In the last few years, the popularity of Transformers (Vaswani et al., 2017) has led to the massive adoption of these neural architectures in several domains, ranging from computer vision (Han et al., 2022) to natural language processing (Brown et al., 2020; Touvron et al., 2023; Chowdhery et al., 2023) and beyond. Nowadays, Large Language Models (LLMs) have gained a significant amount of attention from a wide range of audience, both in the industry and in the academy, due to their outstanding capability of supporting convincing linguistic interactions with humans. For example, in the realm of Question Answering (QA), the pervasive use of LLMs has become a prevailing trend (Kamalloo et al., 2023) dominating vertical tasks through either fine-tuning or integration into more structured learnable models (Hu et al., 2022). Nevertheless, LLMs can occasionally hallucinate by producing plausible, but yet still incorrect answers (Ji et al., 2023; Zhang et al., 2023; Bang et al., 2023). Furthermore, the evolving nature of real-world contexts, where knowledge is dynamically accumulated and updated over time, poses a significant challenge for LLMs. Indeed, retraining or adapting LLMs can both be a cumbersome procedure, requiring specific datasets, and might end-up in introducing forgetting effects, loosing precious information (Parisi et al., 2019).

In order to overcome these limitations, this paper introduces Continual In-context Knowledge LLM (CIK-LLM), a novel method combining LLMs with Temporal Knowledge Graphs (TKGs) that dynamically change over time, possibly in a lifelong manner (De Lange et al., 2021). Departing from the prevalent practice of employing LLMs for both language processing and knowledge management (Zhang et al., 2019; Liu et al., 2021; Kim et al., 2023b), we decou-

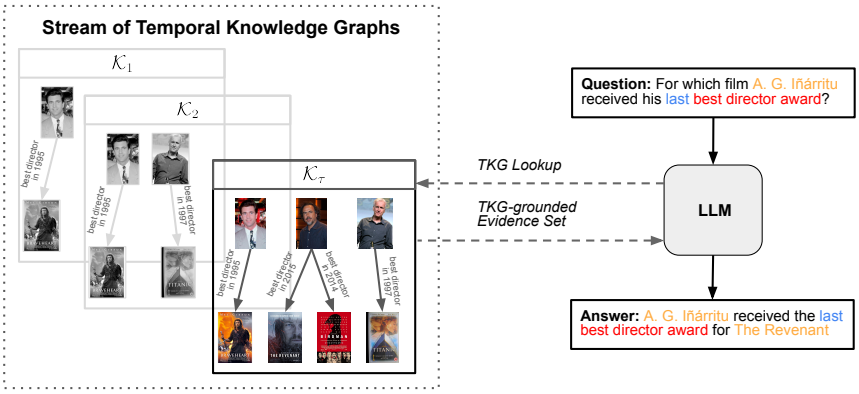


Figure 1: Visual sketch of the scenario in which the proposed CIK-LLM operates. Given a question in natural language, the LLM exploits a dynamically changing TKG (a stream of TKGs) to output a knowledge-grounded answer, possibly involving multi-hop reasoning on it. In the picture we use orange, red and blue to highlight the entities, relations and temporal words, respectively.

ple the roles of language and knowledge in LLMs, delegating the responsibility of continuously storing and updating information to TKGs. The TKG progressively accumulates and updates information over time, while the LLM has the task of spotting relational paths (within the TKG) that align with the input question. Subsequently, our approach identifies subgraphs of evidences, which are encoded in a compressed-and-prompt-friendly manner, seamlessly connecting the LLM and the TKG with the purpose or reacting to the user query. This integration leverages the LLM inherent capabilities to support the reasoning process, leading to more informed and contextually relevant answers, going beyond the knowledge encoded in the weights of the language model. Notably, our approach provides LLMs a natural adaptive capability, which become able to quickly assimilate newly accumulated knowledge, without any further training procedures, thus being an instance of in-context learning (Brown et al., 2020). In particular, CIK-LLM achieves performances that are close to the ones of state-of-the-art models that, despite exploiting LLMs as building blocks of their architecture, require to be trained on extensive datasets of facts, questions, and answers. Moreover, CIK-LLM presents a noteworthy "forgetting-free" approach, allowing adaptation to novel domains without resorting to fine-tuning, QA datasets, or incremental learning procedures. In summary, this paper provides the following contributions:

(i) We define CIK-LLM, a novel approach pairing an LLM with a TKG whose factual knowledge is progressively added/updated over time. The model overcomes the problem of adapting LLMs to handle never-seen-before knowledge, giving the user full control on the way such knowledge is expanded. Our strategy allows an LLM to answer questions in an incremental setting that would require progressive training procedures and adaptations, typical of several continual learning models.

(ii) We propose to handle this problem by means of in-context learning, thus without any retraining or fine-tuning, decoupling the role of storing knowledge (TKG) and the one of handling language, indexing, reasoning on the graph (LLM). Our results show that CIK-LLM is very flexible and it can promptly react to novel knowledge, yielding performances that are compatible with the ones of extensively trained competitors, outperforming them on new knowledge not supported by annotated datasets of questions and answers.

(iii) We propose a novel strategy to map a given question onto a customizable TKG-based symbolic representation, following the route of recent approaches that, however, are not designed to work over time, thus extending them toward our dynamic setting.

(iv) We propose to use a compressed representation of the subgraph which stores the evidence set, exploiting the LLM capabilities of handling symbolic representations that are typical, for example, of programming languages.

This paper is organized as follows. Section 2 introduces background concepts. Section 3 describes our model, while experiments are in Section 4. Section 5 is about related work and Section 6 draws conclusions and future directions.

2 BACKGROUND

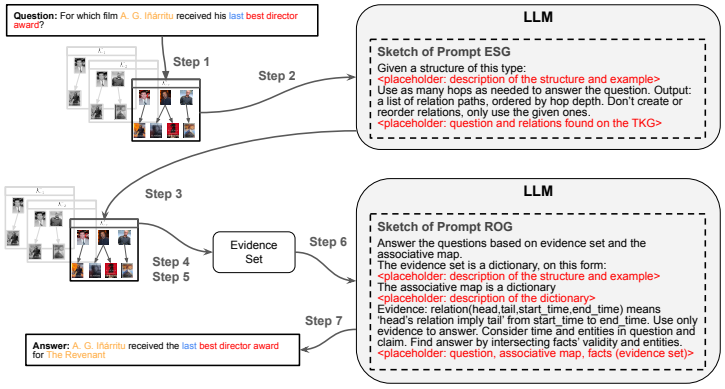
Temporal Knowledge Graphs. A Knowledge Graph (KG) (Hogan et al., 2021) collects a set of entities (e.g., *Birdman*, *A. G. Iñárritu*) along with their interconnected relationships (e.g., *being the director of*) in a graph-based structure representing a set of facts, which are triples composed of two entities and a relation among them (such as *A. G. Iñárritu is the director of Birdman*). More formally, given a set of entities $\mathcal{E}$ and relations $\mathcal{R}$, a KG $\mathcal{G}$ can be defined as a tuple $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denotes the set of facts. Standard KGs are generally defined as a static collection of facts, where triples do not explicitly include a temporal dimension. However, this representation can be often very limiting in real-world scenarios, where the relational interconnections may change as time pass by. For instance, *Birdman is the last film directed by A. G. Iñárritu* denoted a true fact in 2014, which became false since 2015. Temporal Knowledge Graphs (TKG) (Mo et al., 2021) overcome this limitation by referring each fact to a specific timestamp, in the form of a time point (e.g., 1995) or a time interval (e.g., 2014-2015), thus modeling the temporal dynamics in facts (Cai et al., 2023). TKGs can be defined as peculiar extensions of KGs, once we introduce timestamps. In particular, a TKG is a tuple $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{F})$ where $\mathcal{T}$ denotes the set of timestamps and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$ or $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T} \times \mathcal{T}$ in case of facts that happened at specific time instants or in time intervals, respectively. By introducing timestamps, we observe that the initial KG can be regarded as a sequence of snapshots or subgraphs, denoted as $\mathcal{K} = (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{|\mathcal{T}|})$, where each subgraph $\mathcal{G}_\tau$ exclusively encompasses the facts associated with its respective timestamp τ.

Answering Questions with Temporal Knowledge Graphs. Knowledge Graph Question Answering (KGQA) consists in answering natural language questions by means of a KG which is used as reference knowledge base (Zheng et al., 2017; Yani & Krisnadhi, 2021). In the case of temporal KG, the information is associated with timestamps and, thus, answering questions also requires the capability of handling the temporal dimension, and we talk about Temporal KGQA (TKGQA). While in KGQA the answer is usually an entity in the KG, TKGQA assumes that the underlying KG is indeed a TKG, and the answer may be either an entity or a timestamp. For example, a question might seek information about relationships between entities at a specific time point or inquire about changes in the graph over a certain temporal interval. Navigating the temporal dynamics of the TKG to retrieve relevant information which is compatible with the temporal constraints of the question requires techniques that can handle temporal reasoning and temporal dependencies within the knowledge graph. (Saxena et al., 2021; Chen et al., 2022). For instance, while asking *When A. G. Iñárritu has received the best director award for "Birdman"?* may require to consider just a single fact, the question *In which film A. G. Iñárritu has received his last best director award?* would require reasoning across multiple facts at once (Figure 1).

Prompting, In-Context Learning. Brown et al. (2020) demonstrated the outstanding capabilities of LLMs in accomplishing novel tasks without the need of retraining the model. Prompting, within the scope of LLMs, pertains to the formulation of instructions to elicit specific outputs from the model. In the scientific literature, several kinds of prompting strategies have been considered (Yu et al., 2023). Instructions range from simple task definitions to complex prompts that incorporate contextual information. This information can be essential to solve tasks that require data retrieved from textual documents or structured elements, such as KGs. The idea of in-context learning covers a large set of approaches to leverage pre-trained LLMs, ranging from zero to few-shot settings and others (Li, 2023), and it has become strongly popular in the last few years (Brown et al., 2020). For example, in the few-shot case, it consists in enriching the prompt with a small number of demonstrations that are paired with their expected output. Chain-of-thought (Wei et al., 2022) further extends in-context learning by providing examples of all the intermediate reasoning steps necessary to provide the correct answer.

3 CONTINUAL IN-CONTEXT KNOWLEDGE LARGE LANGUAGE MODEL

Despite LLMs have shown excellent reasoning capabilities in performing question answering tasks, they struggle in providing accurate answers in real-world scenarios, where the factual knowledge may change as time pass by. For instance, in the fields of finance and news it is fundamental to rely on the last updated information. In these domains, knowledge changes continuously due to unpredictable events, new data and market fluctuations. This implies that a machine learning model that relies on outdated knowledge could lose accuracy and reliability over time. Two common techniques to mitigate this issues consist in fine-tuning and retraining the model on the novel data. However, both these approaches still have some limitations. Let assume Δ_k is the frequency of knowledge updates over time (e.g. 1 day), and Δ_t the number of times the model can be fine-tuned, or retrained, in the same time interval (on the whole available knowledge every time). If $\Delta_k > \Delta_t$, the knowledge changes faster than the capacity of the model to adapt to it, thus making, e.g., fine-tune useless. The scenario is even worse with retraining, which consists of training the model from scratch with all the available knowledge, involving a higher computational and temporal cost, which may not



Step 1 Retrieve the full collection of relations grouped by distance from each entity.

Step 2 *Build prompt* Π_{esg} to get a list of relational path using the LLM. Check alignment between relation path and $\mathcal{K}_\tau$.

Step 3 Retrieve the subgraph following the identified relational paths.

Step 4 Prune the evidence set (2 stages). Retrieve the verbalized representation of the pruned evidence set.

Step 5 Compress the evidence set (Fig. 3).

Step 6 *Build prompt* Π_{rog} to retrieve the answer using the LLM.

Step 7 Decompress the answer (if needed).

Figure 2: CIK-LLM. For each entity in the given question, we discover relations up to n hops. The LLM chooses the best relation path for each entity (prompt Π_{esg}). Traversing the graph by the found relation paths we get the evidence set, which is pruned and compressed. The LLM infers the answer from the compressed evidence set (prompt Π_{rog}). Finally, we decompress the answer. Prompts are reported with placeholders. A complete example, including Π_{esg} and Π_{rog}, can be found in Appendix (respectively Figure 7 and Listing 1-2).

be sustainable in dynamic and rapidly evolving domains. The problem we consider in this paper is framed in such a dynamic scenario. We formally define the problem and then describe the structure of the proposed model (Section 3.1).

Problem Statement. Standard TKGs assume that the temporal facts can be split into specific timestamps as a sequence of sub-graphs $\mathcal{K} = (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{|\mathcal{T}|})$ (cfr. Section 2), where each $\mathcal{G}_\tau$, for $\tau \in \mathcal{T}$, contains only facts with the timestamp τ. However in our approach, we want to represent a dynamic environment in which the reference temporal graph can grow and update through time. For this reason, we consider a *stream of TKG*, i.e. a (possible infinite) tuple $\mathbb{K} = (\mathcal{K}_1, \mathcal{K}_2, \dots,)$ of TKGs, where each $\mathcal{K}_\tau$ is a TKG containing facts whose temporal coordinate τ' is $1 \leq \tau' \leq \tau$, and we assume that each $\mathcal{K}_\tau$ contains all the facts that are known to be valid at time τ. To convince ourselves this assumption is more appropriate to model real-world scenarios, it is enough to think that before the 5th century A.D. people thought that the sun was revolving around the earth! Luckily humans improve their knowledge as time pass by. Therefore, let us assume $\mathbb{K}$ denotes a TKG stream representing a certain domain knowledge that changes over time. We denote by $\mathcal{K}_\tau$ the snapshot of $\mathbb{K}$ at time $\tau \in \mathcal{T}$, which contains temporal facts $\mathcal{F}_\tau$ until τ that are known to be valid at τ, i.e. the available knowledge at τ. Let $\mathcal{E}_\tau, \mathcal{R}_\tau, \mathcal{T}_\tau$ denote the set of entities, relations and timestamps occurring in some fact of $\mathcal{F}_\tau$, respectively. Then the overall problem can be defined as follows: *given* a natural language question q, a TKG $\mathcal{K}_\tau$ as described above, and a set of entities $\mathcal{E}_q$ in the question, with $\mathcal{E}_q \subseteq \mathcal{E}_\tau$, *produce* an answer a and a justification j in natural language that are consistent with the current state τ of the temporal knowledge graph.

3.1 Model

Motivated by the discussion at the beginning of this section, we propose CIK-LLM[1], a novel method avoiding the continuous fine-tuning of large models. Instead of retraining the LLM to adapt to new facts, we keep the model frozen and leverage its skills in manipulating language and its reasoning capabilities to deal with the continuous updates of the temporal knowledge in the TKG stream. In the following, we present a graph-based reasoning technique for in-context learning, and we introduce a framework that enables a generic LLM to distill the knowledge of the graph given a question q, and answer accordingly with the subgraph obtained from the distillation. Starting from a question q and a set of entities comprised in the question, our approach acts distinct phases to retrieve an answer, ranging from building the evidence subgraph to reason on the set of extracted facts. Figure 2 reports a sketch of the overall process, that will be described in the following.

3.1.1 Retrieval

We are given a temporal knowledge graph $\mathcal{K}_\tau$ consisting of entities $\mathcal{E}_\tau$ and relations $\mathcal{R}_\tau$ associated by facts that are framed in a time interval. Injecting all the facts of $\mathcal{K}_\tau$ into the LLM prompt would be feasible only for small sized knowledge graphs, but impracticable at larger scales since we are limited by the maximum context length of the LLM itself. Thus, we need to retrieve only those facts that are most likely to be relevant to the query at hand. In other

[1] https://github.com/gnekt/llm-and-dtkg

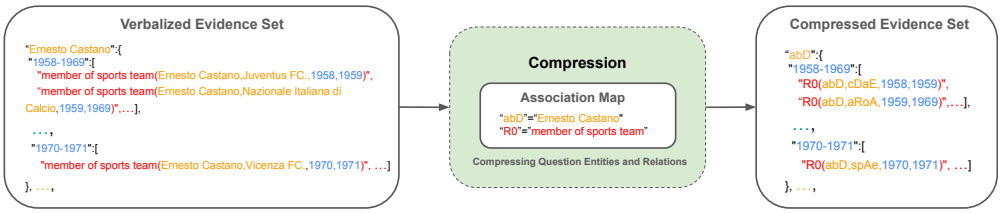


Figure 3: From plain to compressed evidence set. Leftmost block: evidence set as a JSON-like dictionary, with keys composed of question entities (first level), then time intervals (second level). Rest of the sketch: compression returns an associations map to assign compact names to relations/entities, ending up in a compressed evidence set.

words, our goal is to build a subgraph $\mathcal{K}_{q,\tau} \subset K_\tau$. We propose a simple approach to retrieve a compact $\mathcal{K}_{q,\tau}$ that is designed around common considerations about questions on temporal graphs and to use the LLM to avoid extensive search procedures on the graph data.

Building the Relational Path. First of all, we search for a chain of relations that define a valid path along which collect facts that are mostly related to the question. Similarly to (Kim et al., 2023b), we start from the entities marked in the question q, collected in $\mathcal{E}_q$, to drive the selection of the most relevant relations with respect to q. It is important to note that if the entities in question are not provided, this introduces an additional layer of uncertainty. However, this aspect is not considered in our paper, as we adhere to methodologies established by preceding research. In detail, for each entity $e \in \mathcal{E}_q$ we gather all the relations where e is involved in (1-hop). Then we keep navigating $\mathcal{K}_\tau$ to detect all the relations far from e up to at most n-hops. The number of found relations at each hop can be relatively large, and our goal is to exploit the LLM to reduce it. In particular, among all the found relations, we depute the LLM (building an appropriate prompt Π_{esg}) to select the most important ones, aiming at selecting at least one from the relations at distance 1 from e (1-hop), and, in general at most one relation per hop. The selected relations define a path $r_1 \to \ldots \to r_n$ that is then used to traverse $\mathcal{K}_\tau$ starting from e, following at the i-th step the relation r_i. During graph navigation, we collect all the encountered facts $F_{q,e}$ (*evidence set*), that populate the subgraph $\mathcal{K}_{q,\tau}$. Of course, the LLM might return invalid relations, that are not aligned with $\mathcal{K}_\tau$ and which do not yield any collectable facts.

Pruning the Evidence Set. The obtained subgraph $\mathcal{K}_{q,\tau}$ can still be too large to fit in an LLM context. This problem emerges more clearly in temporal graphs, where facts depend on time, that must be provided to the LLM as well. Hence, we further prune $\mathcal{K}_{q,\tau}$ via a two-stage criterion, that ensures that the maximum number of facts after stage k will be $\leq \delta_k$ (customizable), $k \in 1, 2$. After each pruning operation on the evidence set, we reduce $\mathcal{K}_{q,\tau}$ accordingly. While we acknowledge the promising results of learned compact-subgraph retrieval methods (Ju et al., 2022; Zhang et al., 2022a; Jafarzadeh et al., 2022), our approach intentionally exploits a rule-based pruning strategy. This choice emphasizes the simplicity of our solution, aligning with our objective of not necessarily seeking the most optimal tools, but rather straightforward and versatile ones. The *first stage* consists in *multi-hop truncation*, i.e., it is based on the intuition that those facts that are too distant from the entities in the question might be less relevant. Starting from $i = n$ (largest hop), we discard all the facts at hop i, and we repeat until $|F_{q,e}| < \delta_1$. When there are only 1-hop elements, we skip this stage. The *second stage* is based on *time-aware pruning*, where we take into account the temporal validity of each retained fact. The rationale of this stage is that the knowledge needed to answer a question is likely to have occurred in the time span where the entities in the question are valid. For this reason, we restrict our subgraph only to the facts spawned in the time frame intersecting the valid lifetime of the question entities in $\mathcal{E}_q$.[2]. Furthermore, in case $|\mathcal{E}_q| > 1$, we need a more aggressive pruning, since multiple facts are retrieved for each $e \in \mathcal{E}_q$. Thus, considering a group of facts having same relation and tail (the second entity in the triple), we only keep the longest-lasting fact among them.

3.1.2 Inference

Once $\mathcal{K}_{q,\tau}$ is retrieved, we provide all its facts to the LLM by means of a specific LLM prompt (Π_{rog}) and depute the reasoning step to the LLM entirely. Notably, multiple facts occurring at different times may be necessary to answer a certain question. We notice that a fact in temporal KGs is represented with the typical (*head*,*relation*,*tail*) triple enriched by the *time interval* information. We verbalize each fact with the textual representation RELATION(HEAD, TAIL, START, END). Instead of listing all the facts one after another, in Figure 3 we structure them in a JSON-like format, grouping them by entity type and year.

[2]For simplicity, we discard all the facts distant in time more than one year from any fact that involves a question entity.

Prompt Compression. In order to fit all the facts of $\mathcal{K}_{q,\tau}$ in the LLM prompt, pruning might be not enough (unless very small δ_k are chosen). It turns out that there is room for transforming the prompt into a different format that is more compact than the just described one. By construction, the subgraph is composed of many facts involving several repetitions of the question entities, and repeated instances of a few different relations. Creating a prompt to instruct the LLM ends up in a largely verbose description. Moreover, entities and relations are generally split into multiple tokens by the LLM tokenizer, as they are typically composed of multiple words, and, sometimes, their terms are not common enough to appear in the tokenizer vocabulary. Based on such observations, we propose to compress the prompt by replacing all the occurrences of the same entity (or relation) in the evidence set, with a unique *very-short-named* variable. The LLM must be aware of this mapping, thus the variable assignment to relations and entities from $\mathcal{E}_q$ is defined inside the prompt itself (associative map). In such a way, we reduce the number of tokens to store entities and relations in the facts. Consequently, we can increase the number of facts we can encode in the prompt. The prompt is then composed of three elements: (i) question, without entity compression; (ii) associative map; (iii) compressed evidence set. The result of prompt compression is outlined in Figure 3. As a matter of facts, the resulting prompt becomes unreadable to humans. However, LLMs are typically trained on large amounts of code data, thus they exhibit strong capabilities in understanding variables, and our experiments (next Section) confirms such capabilities.

3.1.3 Scalability

Thanks to its retrieve-prune-and-compress pipeline, CIK-LLM is well suited to handle large TKGs. The quality of the result in large-scale contexts mostly depends on the structure of the TKG and the way the information is organized, more than on its size. In fact, when the KG is organized to store fine-grained information for each entity, requiring very long paths/structures to connect most of the question entities to the associated answers, it might happen that the path is too long with respect to the selected n. Moreover, in that specific case, CIK-LMM may end-up in generating a $\mathcal{K}_{q,\tau}$ which does not fit the prompt Π_{rog}, despite the compression. We successfully tested CIK-LMM with graphs of hundreds of thousands of facts and entities.

4 Experiments

We performed experiments using benchmarks for temporal KGQA, both based on existing data and on variants that we specifically created for this setting.

Datasets. CronQuestions (Saxena et al., 2021) is a temporal KGQA benchmark constructed from a subset of Wikidata facts enriched with temporal annotations (Lacroix et al., 2019). The dataset consists of 410k question-answer pairs (indicated with Q) grounded to a TKG having about 328k facts made of 125k entities, 203 relations. Time is expressed in years and each fact is located in a temporal interval. Each question belongs to one of five categories, in function of the role of the temporal component: simple time, simple entity, before/after, first/last or time join. Moreover, each question is paired with the list of mentioned entities. The set of questions Q is divided into train, validation, and test sets splits. We further divide the dataset according to the number of entities present in questions (single/multiple), and the type of answer (time/entity). In order to simulate a dynamic temporal scenario with a stream of TKG, we split the CronQuestions KG $\mathcal{K}$ into three subgraphs, respectively $\mathcal{K}_{<1960}$, $\mathcal{K}_{<2000}$ and $\mathcal{K}_{all}$, such that $\mathcal{K}_{<1960} \subseteq \mathcal{K}_{<2000} \subseteq \mathcal{K}_{all} = \mathcal{K}$, each containing only facts falling into the time interval specified by the subscript (e.g., $\mathcal{K}_{<1960}$ contains all the knowledge until '60). By splitting $\mathcal{K}$ in this manner, we can effectively model and answer questions that require knowledge on how certain facts have changed or remained constant throughout different times in history. We also considered a variant of CronQuestions, that we specifically created to simulate new knowledge that is not memorized in advance from the LLM, whose graph will be denoted by $\mathcal{K}'$, and which has the same temporal and relational structure of $\mathcal{K}$. However, entities are not the same of $\mathcal{K}$ since they were obfuscated by swapping the two halves of each word in the entity label, such as "Ertorob Giobag" in place of "Roberto Baggio". We denote by Q' the set of questions we created (from Q) for $\mathcal{K}'$.

Competitors. We compare CIK-LLM with two recently proposed methods. The first one is TempoQR (Mavromatis et al., 2022), which is designed for TKGQA. It is based on a LLM, a fact encoder, and other neural components that requires retraining or fine-tuning, thus it strongly depends on a training set of question-answer pairs, and not only on a knowledge graph. TempoQR first integrates entity and temporal information in the encoding of the input question given by a pre-trained language model. Subsequently, a score is computed for each entity and timestamp with respect to the enriched question representation. The final answer is then determined by selecting the entity (timestamp) with the highest score. The second competitor is KG-GPT (Kim et al., 2023b), which is fully based on LLMs and, similarly to our approach, it is fully based on in-context learning, without any training operations, even if it was not proposed in the context of TKGQA. KG-GPT relies on a pre-trained LLM to split the input question into sub-sentences, select

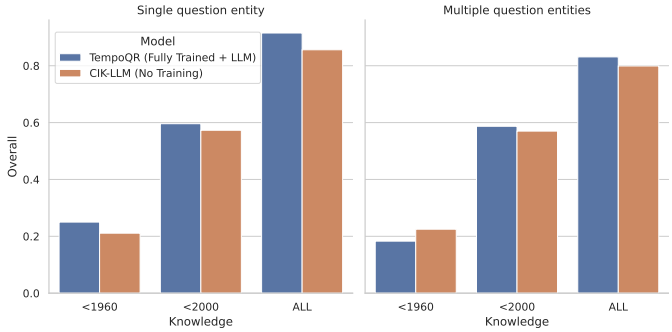


Figure 4: Overall (i.e., all question types) hit@1 score, comparing CIK-LLM (no training) and TempoQR (trained on a large set of QAs and exploiting an LLM as well) in an incremental knowledge environment. CIK-LLM reaches results that are not far from TempoQR, *without relying on any training procedures*.

Table 1: Hits@1 scores in (S)imple/(C)omplex question forms (Q.) and time/entity answer (A.). Left: single question entity. Right: multiple question entities.

Type	Model	$\mathcal{K}_{<1960}$	$\mathcal{K}_{<2000}$	$\mathcal{K}_{all}$
Q - S	TempoQR	0.247	0.665	0.988
	CIK-LLM	0.210	0.581	0.874
Q - C	TempoQR	0.255	0.498	0.810
	CIK-LLM	0.213	0.562	0.830
A - Time	TempoQR	0.067	0.279	0.793
	CIK-LLM	0.179	0.520	0.760
A - Entity	TempoQR	0.272	0.633	0.929
	CIK-LLM	0.215	0.580	0.867

Type	Model	$\mathcal{K}_{<1960}$	$\mathcal{K}_{<2000}$	$\mathcal{K}_{all}$
Q - S	TempoQR	0.251	0.707	0.970
	CIK-LLM	0.248	0.675	0.926
Q - C	TempoQR	0.154	0.535	0.770
	CIK-LLM	0.215	0.525	0.743
A - Time	TempoQR	0.218	0.636	0.908
	CIK-LLM	0.264	0.694	0.928
A - Entity	TempoQR	0.161	0.555	0.780
	CIK-LLM	0.200	0.490	0.715

the most relevant relation for each sub-sentence, and then answer by traversing the graph with a set of collected facts for each sub-sentence-relation pair.

Setup. We selected $5k$ questions from the test sets of Q and Q', composed of 1k questions from each question category. To get a fair comparison, both CIK-LLM and KG-GPT use the same LLM backbone, which is SolarM-SakuraSolar-SLERP,[3] a 10.7B parameters model derived from merging Solar (Kim et al., 2023a) and Sakura[4] that is a tuned version of Solar on math-related instructions eliciting reasoning capabilities of the model. At the moment we are writing, SolarM-SakuraSolar-SLERP is highly ranked among similarly sized LLMs in hugginface leaderboard.[5] The pruning thresholds are set to $\delta_1 = 300$, while $\delta_2 = 30$. The maximum number of allowed hops is $n = 3$. To evaluate TempoQR, we used the same hyper-parameters as the original paper and trained it exploiting the different subgraphs of $\mathcal{K}$ we considered in our experience. We modified the prompt of KG-GPT to support temporal questions, by adding an explanation of how time is represented and how the temporal validity of each fact is defined. All the reported results are based on the hits@1 score, which indicates whether the correct answer (time or entity) is ranked first in the label set proposed by the model.

4.1 Results and Key Findings

Main Results on the TKG Stream. Figure 4 reports the main results in the case of the three evolving temporal knowledge graphs, for the case of single and multiple question entities. The proposed CIK-LLM is capable of achieving results that are not far from the ones of TempoQR in all the scenarios, without requiring any training, while TempoQR is a dedicated model trained on the downstream task itself. This result is further investigated in Table 1, where different types of questions are considered. Our model reaches comparable performance as TempoQR in almost all the categories. Moreover, we notice a significant improvement for single entity questions in which the answer type is a time over $\mathcal{K}_{<1960}$ and $\mathcal{K}_{<2000}$.

[3] huggingface.co/kodonho/SolarM-SakuraSolar-SLERP
[4] huggingface.co/kyujinpy/Sakura-SOLAR-Instruct-DPO-v2
[5] huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

In-depth Study & **Comparisons.** To better emphasize CIK-LLM capabilities, In Table 2 we show comparisons with other approaches and with instances on CIK-LLM without the prompt compression strategy (Section 3.1). In particular, we considered the case in which we simply directly provide the questions to the LLM as is, without providing any additional knowledge, in this way we can clearly investigate to which extent the LLM alone already includes knowledge about the test questions, gained during its pre-training stage. We considered GPT-3.5-turbo and Sakura (GPT-plain/Sakura-plain). The results show that knowledge injection significantly enhances the LLM capability of retrieving the correct answer, since the original knowledge of the plain LLMs is not sufficient to answer questions that require specific or uncommon knowledge or facts. KG-GPT performs slightly better on single entity questions than CIK-LLM, but it is outperformed on multiple entity questions. This is because the sentence splitting procedure in KG-GPT fails to properly segment a sentence when there are multiple question entities, resulting in errors in knowledge extraction from the KG. As expected, TempoQR outperforms all the other models in both single and multiple scenarios, being it fully trained on the question-answer pairs of this benchmark. Regarding the percentage of answered questions (reported in brackets in Table 2), TempoQR/GPT-Plain/Sakura-Plain achieve a 100% ratio because they do not encounter any graph traversal or token length limit errors. With KG-GPT and our solution, the ratio is slightly lower than 100% for different reasons: KG-GPT has difficulties in parsing LLM outputs when the LLM is not designed to output structured text or to split complex questions, whereas our solution faces token limit issues when the number of facts increases.

Table 2: Comparison of different methods, hits@1 score. The results are calculated only on questions which are correctly parsed (whose percentage is reported in brackets in the table)

Model	$\mathcal{K}_{all}$ - Single Q	$\mathcal{K}_{all}$ - Multiple Q
TempoQR	0.915 (100.0%)	0.831 (100.0%)
KG-GPT	0.871 (96.88%)	0.575 (97.05%)
GPT-Plain	0.159 (100.0%)	0.346 (100.0%)
Sakura-Plain	0.136 (100.0%)	0.334 (100.0%)
CIK-LLM w/o compression	0.890 (98.53%)	0.797 (95.45%)
CIK-LLM	0.870 (98.35%)	0.819 (97.51%)

Impact of Pruning. We measure the effectiveness of our pruning strategy by comparing the number of facts and the token length of prompt Π_{rog} before and after pruning (Table 3). We find that our pruning strategy dramatically reduces both the number of facts and the prompt length (in tokens) by almost two order of magnitudes. Without pruning, the retrieved knowledge would not have fit in the prompt in about 48% of the questions.

Impact of Prompt Compression. In this experiment, we assess the impact of compressing the evidence set in terms of both performance and prompt reduction. In Figure 5, we illustrate the relationship between sequence length and the number of facts in the prompt. As expected, the bigger is our evidence set, the more compression is effective. The results presented in Table 2 reveal that compression positively impacts performance when the question involves multiple entities. In such scenarios, the knowledge graph retrieved contains a greater number of facts, and compression enables the inclusion of more information in the prompt. Conversely, for questions pertaining to a single entity, it is typically feasible to accommodate all facts within the prompt even without compression. In such instances, a compressed prompt may slightly diminish performance. However, it is noteworthy how proficiently LLMs can interpret such compressed representations of facts not understandable by humans.

Table 3: Number of facts and number of tokens of the retrieved subgraph, before and after the two pruning stages.

	#Facts (avg)	#Facts (std)
Initial Retrieval	902	2,635
Multi-hop Truncation	261	320
Time-Aware Pruning	21	36
	#Tokens (avg)	#Tokens (std)
Initial Retrieval	33,620	92,480
Multi-hop Truncation	11,060	11,190
Time-Aware Pruning	2,610	1,329

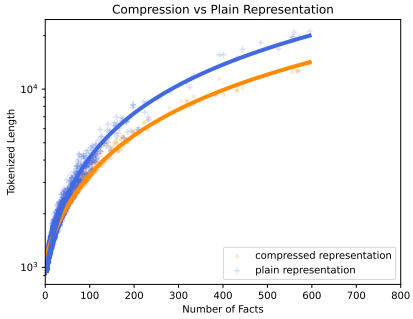
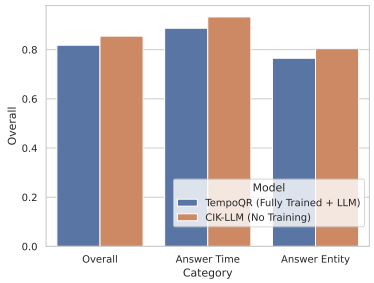
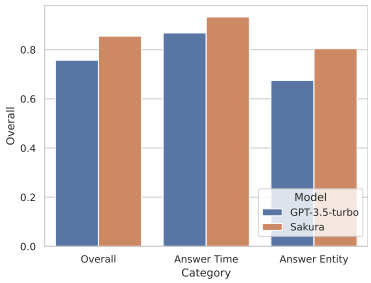


Figure 5: Tokenization length in function of the number of facts in the prompt, with and w/o prompt compression.

Comparison with a different TKG. To further explore CIK-LLM's ability to handle knowledge updates, we conduct a comparative analysis using as knowledge a merged version between $\mathcal{K}'$ and $\mathcal{K}_{all}$. To assess the capacity of TempoQR and our solution to a cold start scenario, where the model knows facts representing the knowledge but it has not the opportunity to be trained on question-answers about the new knowledge, we use the training set of Q (only needed by TempoQR), while we measured performance on test set from Q'. This means that we are going to ask questions about knowledge in $\mathcal{K}'$, without providing training question-answer pairs about such piece of knowledge. Of course, in order to perform a fair comparison with TempoQR in terms of knowledge utilization, we retrained the inner module of TempoQR that encodes entities and facts (TKGE module), to make it aware of the extended knowledge graph $\mathcal{K}_{all} \cup \mathcal{K}'$. In Figure 6a, CIK-LLM consistently outperforms TempoQR across all metrics (when using the LLM alone/plain, no right answers are provided to Q'). This superiority arises because TempoQR requires a dedicated training set of questions specifically related to the new knowledge in order to surpass our performance. Consequently, we assert that TempoQR falls short of our solution in a cold start scenario. In Figure 6b, we further compare with GPT-3.5-turbo and Sakura. Notably, Sakura demonstrates superior performance over GPT-3.5. This advantage can be attributed to Sakura's training on code and math datasets, enabling it to better handle variable associations introduced by the compression prompt.



(a) CIK-LLM vs TempoQR



(b) CIK-LLM: Sakura vs GPT-3.5T

Figure 6: Experimenting with the additional knowledge graph $\mathcal{K}'$ on questions Q' (i.e., new questions about knowledge which is fully out of the scope of the LLM). (a): CIK-LLM and TempoQR; (b): CIK-LLM exploiting Sakura or GPT-3.5-turbo. For a fair comparison with TempoQR, we trained the inner module TKGE of TempoQR using also the new knowledge, i.e., $\mathcal{K}_{all} \cup \mathcal{K}'$.

5 Related Work

In the following we describe those works which are mostly intersecting the ideas of this paper, emphasizing the way in which they are related to our work and what is novel in our proposal.

Knowledge-Augmented LLMs. Integrating LLMs and KGs to enhance their respective strengths is an increasingly popular trend. Pan et al. (2024) categorizes the integration methodologies into three main types: (i) KG-enhanced LLMs, (ii) LLM-augmented KGs, and (iii) synergized LLMs with KGs. While (ii) and (iii) have received some attention by different authors (Yao et al., 2019; Jiang et al., 2022; Yasunaga et al., 2022; Choudhary & Reddy, 2023), the approach considered in this paper is mostly related to category (i). Enhancing LLMs with KGs provides advantages such as accurate explicit knowledge and support for symbolic reasoning (Chen et al., 2020; Zhang et al., 2021; Toroghi et al., 2024). Indeed LLMs, trained on extensive unsupervised data, often lack precise domain knowledge, making KGs valuable for supplementing this deficiency. KGs also offer the advantage of being continuously updated to represent a dynamic learning environment (Mitchell et al., 2018). In general, integration of KGs with LLMs can occur during training or inference and serves interpretability purposes. Approaches applying KGs during the pre-training stage enable LLMs to leverage linguistic and semantic contexts for optimal performance in downstream tasks. For instance, the KG can be integrated into the training objective (Liu et al., 2020), into the LLM input (Zhang et al., 2022b), or used for fine-tuning (Wang et al., 2022; Ye et al., 2022). However, real-world knowledge is dynamic and requires frequent updates, but pre-trained methods do not have such flexibility and would require model retraining. An alternative is injecting KGs during the inference step. In recent years, Retrieval-Augmented Generation (RAG) methods have become very popular, i.e., methods that aim to improve LLMs performance by retrieving additional information and injecting it in the prompt. From a generic perspective, our work can also be considered as a particular instance of this broad class of methods. The majority of RAG approaches retrieve the knowledge from either an external textual source or, less frequently, from a structured source (e.g., graphs). For instance, Guu et al. (2020); Lewis et al. (2020) retrieve relevant knowledge from a large corpus and provide it to the LLM during inference, Shi et al. (2023) use a tuneable retrieval model, Zhang et al. (2024) propose a way to filter out noisy documents obtained during retrieval, and Jiang

et al. (2022) unifies fact retrieval and reasoning into a single framework. On the other hand, when the knowledge is structured in a KG, RAG models face the retrieval problem in a different ways. For instance, Logan et al. (2019) select the facts from the KG using the current context to generate factual sentences, Luo et al. (2024) generate all possible reasoning paths which lead to useful information to be added in the prompt context, He et al. (2024) use a combination of LLMs and GNNs to capture the most appropriate sub-graph given a certain question. However, despite the rapid growth of works aiming at enhancing LLMs using RAG over KGs, none of the existing methodologies have been designed to deal with a scenario in which temporal aspects are incorporated into the knowledge, and hence trained or tuned retriever may drastically suffer of forgetting as knowledge increase through time. Last but not least, a simple yet effective way to inject the information contained in a KG in a LLM is by prompt engineering, which has the advantage of using the LLM without any fine-tuning, while leveraging the structure of KGs to perform reasoning. For instance, Li et al. (2023) converts triplets into short sentences which can be understood by LLMs for reasoning, and (Luo et al., 2023) samples different relation paths from KGs which are verbalized and fed into the LLM. Our approach is very related to these last discussed methods. However none of them has never considered the case of temporal KGs, where the knowledge and reasoning techniques are explicitly time-depending.

Temporal KGQA. The majority of approaches based on TKGs adopt embedding representations for entities, relations and timestamps in a low-dimensional vector space (Dasgupta et al., 2018; Lacroix et al., 2019), mostly focusing on KG completion tasks. Saxena et al. (2021) take a step forward to investigate TKGQA by introducing the dataset CronQuestions and the model CronKGQA, which exploits TKG embedding methods like TComplEx (Lacroix et al., 2019) to efficiently answer simple queries as in a link prediction task over the TKG. However, CronKGQA performances on complex questions requiring to consider multiple facts to get a correct answer were quite unsatisfactory. Enhanced approaches like TempoQR (Mavromatis et al., 2022) and SugGTR (Chen et al., 2022) turned out to be more effective in addressing complex queries either by augmenting the question embeddings with additional context and time-aware information, or by first retrieving a set of temporal facts and then perform temporal reasoning on the TKG. All these models expolit a language model to extract natural language question embeddings and combine it with temporal embeddings to perform QA over the TKG. Our approach significantly differs from these models as the QA task is directly solved by the LLM without retraining. We use the TKG solely to bridge on-the-fly the LLM with the available set of facts, possibly evolving through time, and not to train enhanced embedding representations.

Continual/Lifelong Learning. The ability to learn without forgetting (Kirkpatrick et al., 2017) is a human capability that Continual Learning approaches (Wang et al., 2024) try to mimic, trying to overcome the widely known issues of neural networks (Parisi et al., 2019). Pre-trained Large Language Models (Brown et al., 2020) not only yield a strong transfer of knowledge when fine-tuning to downstream tasks, but their capabilities also allow to learn new tasks by simply specifying proper instructions to the model without the need of any training. That is why in NLP many continual learning approaches focus on instruction-based methods (Ke & Liu, 2022). In prefix-tuning (Li & Liang, 2021) a frozen Language Model is adapted to new tasks by simply optimizing task-specific vectors that are prepended to the input. Similarly, Lester et al. (2021) learn k task-specific tokens per task. In Zhu et al. (2022), prompt tuning vectors are learnt in such a way to allow knowledge transfer between different tasks. ConTinTin (Yin et al., 2022) learns a different instruction for each task in an incremental setting. The benchmarks of (Jang et al., 2022) are based on text corpora with emphasis on continuous model training. In this paper instead, we do not involve any learning of the LLM. Instead, we depute TKGs to deal with the continual dynamic changes of the world.

6 Conclusions and Future Work

This paper introduced Continual In-context Knowledge Large Language Models (CIK-LLM), a novel approach combining LLMs inference capabilities with a stream of temporal knowledge graphs. In the experimental analysis, we showed that our method has similar performances to models that are trained on specific data, but without requiring any training procedure. Moreover, one advantage of CIK-LLM is its local exploration of the given TKG, ensuring that CIK-LLM remains scalable and applicable even when dealing with larger KGs. In addition, CIK-LLM outperformed state-of-the-art models like GPT-3.5-turbo on uncommon knowledge tasks, and it is more robust to cold start scenarios with respect to pre-trained models. Overall CIK-LLM represents a simple and flexible approach to face problems with dynamic knowledge that evolves over time. A possible limitation of our approach is that the relational paths highlighted by the LLM could not be compatible with the provided KG. This is mostly due to the fact that we do not specify in Π_{esg} how the relations at different depths are linked to each other. On the other hand, this allows us to define a simpler prompt for the LLM. This research demonstrates how to combine the reasoning capabilities of Large Language Models with a continual stream of knowledge changing through time. This has the main advantage of enabling a more transparent and accurate reasoning capability in the model, and prevent invalid inferences and hallucinations in LLMs. We think that a further step forward in this direction, would be to let an LLM deal with multiple sources of information at once, e.g., a set of TKGs (possibly not fully coherent one to one another). In future work we also plan to investigate the case of multiple knowledge-enriched LLMs cooperating to solve a common objective.

ACKNOWLEDGMENTS

This study was supported by TAILOR project funded by EU Horizon 2020 under GA No 952215. This work has been also supported by the Partnership Extended PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI". The acknowledgment is extended to Rete SAIHUB (Siena, Italy), which, jointly with the Italian Ministry of University and Research (DM 117/2023, PNRR, Missione 4, Componente 2, Investimento 3.3), funded the scholarship of Christian Di Maio.

REFERENCES

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. Temporal knowledge graph completion: A survey. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 6545–6553. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/734. URL https://doi.org/10.24963/ijcai.2023/734. Survey Track.

Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

Ziyang Chen, Xiang Zhao, Jinzhi Liao, Xinyi Li, and Evangelos Kanoulas. Temporal knowledge graph question answering via subgraph reasoning. *Knowledge-Based Systems*, 251:109134, 2022.

Nurendra Choudhary and Chandan K Reddy. Complex logical reasoning over knowledge graphs using large language models. *arXiv preprint arXiv:2305.01157*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2001–2011, 2018.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, 2024.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Parastoo Jafarzadeh, Zahra Amirmahani, and Faezeh Ensan. Learning to rank knowledge subgraph nodes for entity retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 2519–2523, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531888. URL https://doi.org/10.1145/3477495.3531888.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models, 2022.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*, 2022.

Mingxuan Ju, Wenhao Yu, Tong Zhao, Chuxu Zhang, and Yanfang Ye. Grape: Knowledge graph enhanced passage reader for open-domain question answering. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 169–181, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.13. URL https://aclanthology.org/2022.findings-emnlp.13.

Ehsan Kamalloo, Nouha Dziri, Charles LA Clarke, and Davood Rafiei. Evaluating open-domain question answering in the era of large language models. *arXiv preprint arXiv:2305.06984*, 2023.

Zixuan Ke and Bing Liu. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*, 2022.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth upscaling. *arXiv preprint arXiv:2312.15166*, 2023a.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023b.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*, 2019.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. Graph reasoning for question answering with triplet retrieval. *arXiv preprint arXiv:2305.18742*, 2023.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Yinheng Li. A practical survey on zero-shot prompt design for in-context learning. In Ruslan Mitkov and Galia Angelova (eds.), *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pp. 641–647, Varna, Bulgaria, September 2023. INCOMA Ltd., Shoumen, Bulgaria. URL https://aclanthology.org/2023.ranlp-1.69.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 2901–2908, 2020.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6418–6425, 2021.

Robert Logan, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5962–5971, 2019.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *arXiv preprint arXiv:2309.01538*, 2023.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning, 2024.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N Ioannidis, Adesoji Adeshina, Phillip R Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. Tempoqr: temporal question reasoning over knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 5825–5833, 2022.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5): 103–115, 2018.

Chong Mo, Ye Wang, Yan Jia, and Qing Liao. Survey on temporal knowledge graph. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, pp. 294–300. IEEE, 2021.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

A Saxena, S Chakrabarti, and P Talukdar. Question answering over temporal knowledge graphs. In *ACL-IJCNLP 2021-59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 6663–6676. Association for Computational Linguistics (ACL), 2021.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

Armin Toroghi, Willis Guo, Mohammad Mahdi Abdollah Pour, and Scott Sanner. Right for right reasons: Large language models for verifiable commonsense knowledge graph question answering, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jianing Wang, Wenkang Huang, Minghui Qiu, Qiuhui Shi, Hongbin Wang, Xiang Li, and Ming Gao. Knowledge prompting in pre-trained language model for natural language understanding. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3164–3177, 2022.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Mohammad Yani and Adila Alfa Krisnadhi. Challenges, techniques, and trends of simple knowledge graph question answering: a survey. *Information*, 12(7):271, 2021.

Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, 2019.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323, 2022.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. Ontology-enhanced prompt-tuning for few-shot learning. In *Proceedings of the ACM Web Conference 2022*, pp. 778–787, 2022.

Wenpeng Yin, Jia Li, and Caiming Xiong. Contintin: Continual learning from task instructions. *arXiv preprint arXiv:2203.08512*, 2022.

Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. Towards better chain-of-thought prompting strategies: A survey. *arXiv preprint arXiv:2310.04959*, 2023.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35, 2021.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5773–5784, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.396. URL https://aclanthology.org/2022.acl-long.396.

Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. Dkplm: decomposable knowledge-enhanced pre-trained language model for natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11703–11711, 2022b.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. Raft: Adapting language model to domain specific rag. 2024.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

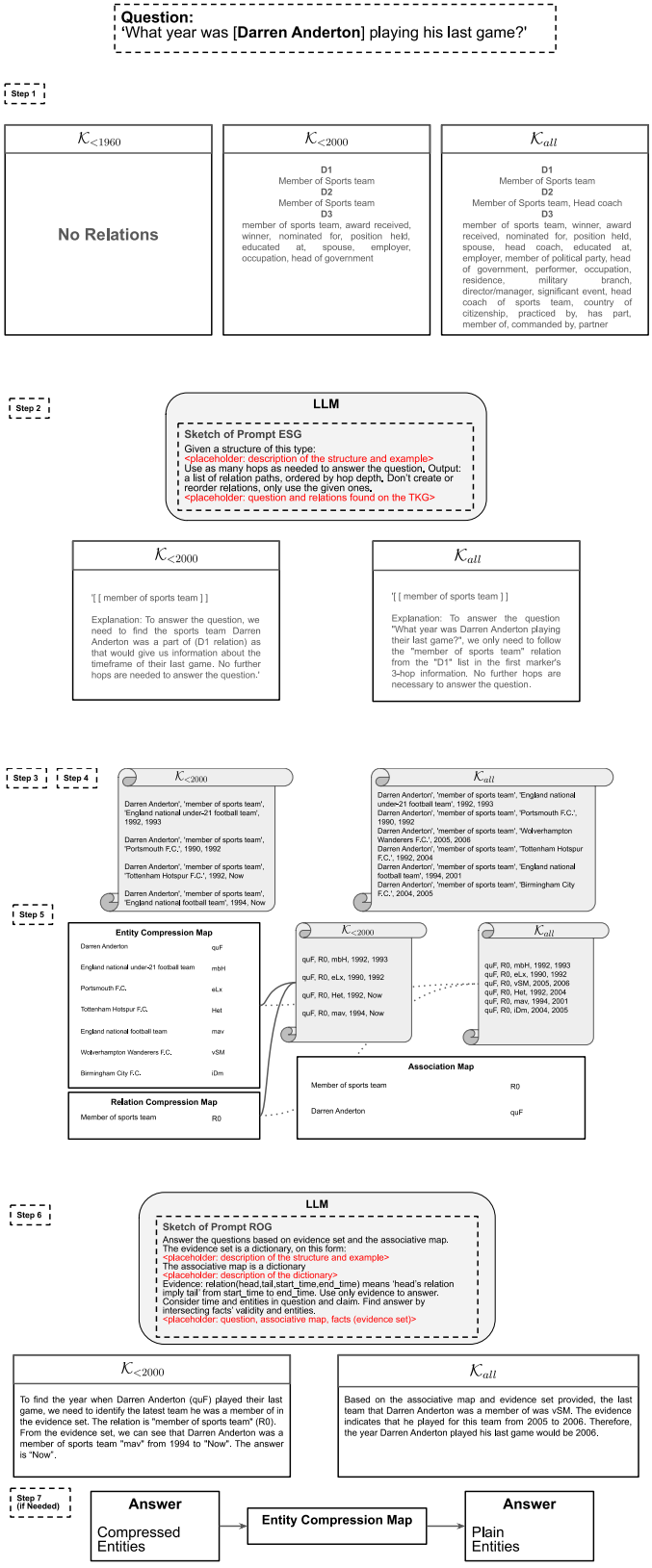
Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1441–1451, 2019.

Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. Natural language question/answering: Let users talk with the knowledge graph. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 217–226, 2017.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. Continual prompt tuning for dialog state tracking. *arXiv preprint arXiv:2203.06654*, 2022.

A RUNNING EXAMPLE

We report a complete running example in Figure 7, consider a real case, and following the steps indicated in Figure 2. This example illustrates that the knowledge has a substantial effect on the answer provided by the model. Indeed, as time advances, the correct answer might change.



Step 1: Given a question, retrieve the collection of key relations, grouped by distance from each entity. We consider a knowledge graph which evolves over time, showing three snapshots ($\mathcal{K}_{<1960}, \mathcal{K}_{<2000}, \mathcal{K}_{all}$) at three different time instants. In $\mathcal{K}_{<1960}$ there is no information about the entity *Darren Anderton*, so the answering process will stop with this message "There's no information in the knowledge about Darren Anderton" (thus we avoid considering $\mathcal{K}_{<1960}$ in what follows).

Step 2: We provide Π_{esg} to the LLM (the complete version can be seen in Listing 1), to extract the most pertinent relational paths for answering the question. In this example, the LLM identifies a single relational path, composed of a single relation.

Step 3-4: Collect the evidence set along the designated relational path, refine this set through the pruning phases explained in Sec. 3.1.1.

Step 5: Gather all entities and relations within the evidence set, then obtain their compressed forms from the *Entity Compression Map* and *Relation Compression Map*. Compressed representations are used in the compressed evidence set. The mappings of question entities and relations are also included in the Association Map.

Step 6: We provide Π_{rog} to the LLM (refer to Listing 2 for the complete version), to obtain the answer. In this example (see the evidence sets in Step 3-4 above), we can see that using $\mathcal{K}_{2000}$ Darren Anderton appears to be actively playing, whereas $\mathcal{K}_{all}$ indicates that he retired in 2006.

Step 7: If the answer pertains to an entity, we must perform a decompression operation (recall that the information managed by the LLM is in compressed form). This step will retrieve the correct answer. Conversely, if the answer does not concern an entity, decompression is not required.

Figure 7: A real example that demonstrates how the answer varies with the use of a Temporal Knowledge Graph (TKG) that changes over time, following the structure of Figure 2.

B PROMPTS

We report an example of full prompts Π_{esg} and Π_{rog}, to provide the reader with a real, placeholder-free instance of the structure we sketched in Figure 2. The former is reported in Listing 1, while the latter is in Listing 2. Of course, they are harder to follow than the example we provided in the main text, but they are useful to the reader who is interested in getting into the details of the specific information we provided to the LLM.

Listing 1: Π_{esg} prompt. The text is about a real example, and its structure represents an expanded version the one shown in Figure 2 (top), using the same color scheme and fully avoiding placeholders.

```
Given a structure of this type:
{"question": "<<<QUESTION>>>",
 "markers": [{"starting_node":"<<<STARTING_NODE_1>>>",
              "3-hop": {"D1": [relation1, relation2, relationN],
                        "D2": [relation1, relation2, relationN],
                        "D3": [relation1, relation2, relationN]}},
             {"starting_node": "<<<STARTING_NODE_2>>>",
              "3-hop": {"D1": [relation1, relation2, relationN],
                        "D2": [relation1, relation2, relationN],
                        "D3": [relation1, relation2, relationN]}}]
Markers contains a list when each item contain a starting_node to access into a knowledge graph and a 3-hop
relation set. The 3-hop field contains relations that we can use to explore the graph starting from the
starting_node.For each hop we can use only one relation, you must select the relation that best express the
question.
Examples:
1)"question": "[Pavel Andreyevich Taran] got [Gold Star] in what year.",
"markers": [{"starting_node": "Pavel Andreyevich Taran",
              "3-hop": "{D1: [award received],
                         D2: [ award received, winner],
                         D3: [award received, military rank, winner, educated at, director/manager,
                          spouse, position held, employer, academic degree, commanded by, country,
                         located in the administrative territorial entity, work location, territory
                         claimed by, residence, capital of, twinned administrative body, twinned
                         administrative body, head of government, significant event, country of
                         citizenship, nominated for, commander of, military branch, chairperson, head
                         coach, work location, convicted of, country of origin, head of government,
                          rector, employer, educated at, member of sports team, named after, member of,
                         residence, officeholder, described by source, followed by, doctoral advisor,
                          doctoral advisor]}",
             {"starting_node": "Gold Star",
              "3-hop": "{D1: [award received],
                         D2: [winner],
                         D3: []"}]}
Expected output: [["award received","winner",""],["award received"]]
Constraints:
1. You don't need to arrive to the 3rd hop if you think that the 1st or the 2nd is enough to answer the
question.
2. Each hop must contain only one relation.
3. Each selected relation must be picked from the appropiate list.
4. The expected output is a list of list of relations, each list of relations is a possible path to answer
the question.
5. Each list of relations must be ordered by hop (D1-D2-D3).
6. If there are 3 markers you need to generate 3 relation list (one for each marker), if there are 2 markers
    you need to generate 2 relation list (one for each marker), if there is 1 marker you need to generate 1
    relation list (one for each marker).
7. Each relation list is placed in the list following the order of the markers.
    Example)\n1) [["award received","winner",""],["award received"], ...]
8. You can't invent relations or change orderings, you must pick relations from the list and respect the
    order, because i need alignment among D1-D2-D3 relations and what you generate.
9. If there are 3 markers you need to generate 3 relation list (one for each marker), if there are 2
    markers you need to generate 2 relation list (one for each marker), if there is 1 marker you need to
    generate 1 relation list (one for each marker).

Now it's your turn.
2)"question": "What year was [Darren Anderton] playing his last game?",
 "markers": [{"starting_node": "Darren Anderton",
              "3-hop": {"D1": ["member of sports team"],
                        "D2": ["member of sports team", "head coach"],
                        "D3": ["member of sports team", "winner", "award received", "nominated for",
                        "position held", "spouse", "head coach", "educated at", "employer",
                        "member of political party", "head of government", "performer",
                        "occupation", "residence", "military branch", "director/manager",
                        "significant event", "head coach of sports team",
                        "country of citizenship", "practiced by", "has part", "member of",
                        "commanded by", "partner"]}}]
 Expected output:
```

Listing 2: Π_{rog} prompt. The text is about a real example, and its structure represents an expanded version the one shown in Figure 2 (bottom), using the same color scheme and fully avoiding placeholders.

```
Answer the questions based on evidence set and the associative map.
The evidence set is a dictionary, on this form:
{ "EntityX":{"start_time-end_time":
["relation(EntityA,EntityZ,start_time1,end_time1)",
"relation(EntityP,EntityY,start_time2,end_time2)",
"relation(EntityO,EntityC,start_time3,end_time3)"]} ...}
EntityX is a subject of the question, "start_time-end_time" is the interval of validity of
the evidence list on it and the list is a list of evidence in a prolog format.
The associative map is a dictionary
where the key is a subject and the value is a variable used to reduce length of the subject.
The associative map gives a short representation for the entity in the question, where the key is a variable
used to reduce length of the subject and the value is the subject.
Example:
Associative map:  X"Ernesto Castano"; Y"member of sports team".  X is a variable used to reduce length of
the subject "Ernesto Castano" and Y is a variable used to reduce length of the subject "member of sports team".
end_time and start_time are temporal informations.
Each evidence is in the form of "relation(head,tail,start_time,end_time)" and it means: "From the year given in
start_time to the year given in end_time, head's relation imply tail".
    Example: Associative map: abX="Ernesto Castano"; R0="member of sports team".
             Evidence: {1958: ["R0(abX,eeW,1958,1959)"]} means that Ernesto Castano was a member of sports team
                       eeW from 1958 to 1959.
The order in which keys are in the evidence set dictionary represent a temporal axis, so they are ordered in
an increasing manner.
The evidence set is the knowledge that you can use, you can base your answer only on these information.
The answer must be always find in the evidence set, if you can't find it directly with the given question, try
to paraphrase it to match the information in the evidence set.
    Example: "What's the last team Ernesto Castano played?" since "played" is not a relation in the evidence set,
             you must find a synonym of "played" that is in the evidence set, in this case
             "member of sports team".
    Example: "Who was the employer of [Magdalena Jetelova] in 2005" since "employer" is a relation in the
             evidence set, you can use it directly.
Questions may be related to temporal aspect of the claim, so you must use the temporal information in the
claim(time or adj or adv) and in the evidence set.
If you can't find the answer, you can consider also that the
interval validity of each fact can fall in the interval validity of another fact.
If some year is proposed in the question, you must use it to access in the right interval for each entity key
in the evidence set.
Remember that time in this case is only expressed as year, so to do operation among time to analyze correctly
the evidence set you can consider them as integer.Of course you can intersect the intervals of validity of
the evidence set to find the answer, moreover you can intersect also the main entities in the question.
Example)
1)Question: What's the last team [Ernesto Castano] was on?
Associative map: X="Ernesto Castano"; Y="member of sports team"
Evidence set: {"X":{"1958-1969": [["Y(X,Q234234,1958,1959)"],
                                  ["Y(X,Q11,1959,1969)"]],
                   "1970-1971": ["Y(X,Q34323,1970,1971)"]"}}
Answer: Since X="Ernesto Castano" and Y="member of sports team", the last team that Ernesto Castano
was on is Q34323.
The Answer is a plain text, so you can write it in any way you want, but it must be a string.
Now it's your turn!
Question:  What year was [Darren Anderton] playing his last game?
Associative map:  quF=Darren Anderton; R0=member of sports team;
Evidence set:  {"quF":  {"1990-2005":  ["R0(quF,eLx,1990,1992)", "R0(quF,Het,1992,2004)",
"R0(quF,mbH,1992,1993)", "R0(quF,mav,1994,2001)", "R0(quF,iDm,2004,2005)"],
"2005-2006":  ["R0(quF,vSM,2005,2006)"]}}
Answer:
```