Unifying Paper

Links

Unifying_Large_Language_Models_and_Knowledge_Graphs_A_Roadmap.pdf

Other sources to look at

- ◯ Building KGs https://ieeexplore.ieee.org/document/9231227 [95]
- ◯ AutoKG - Efficient Automated Knowledge Graph Generation for Large Language Models https://github.com/wispcarey/AutoKG (KG Completion), *can be used to design several prompts for different KG construction tasks* (e.g., entity typing, entity linking, and relation extraction) by PiVE [158]
- ◯ LLMs as Agents Reasoning - later maybe if we continue with retrieval

---

Problems

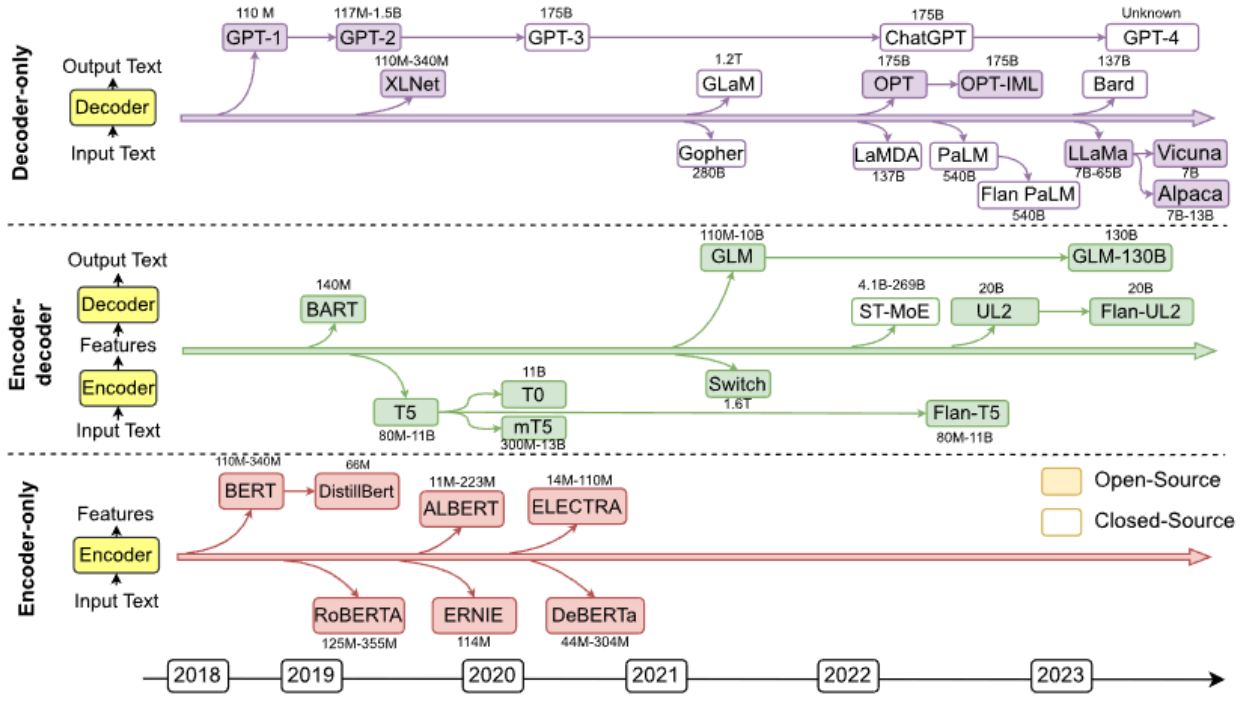
LLMs

- LLMs trained on general corpus might not be able to generalize well to specific domains or new knowledge due to the lack of domain-specific knowledge or new training data

KGs

- KGs are difficult to construct
- Current approaches in KGs [27], [33], [34] are inadequate in handling the incomplete and dynamically changing nature of real-world KG
- In addition, they often ignore the abundant textual information in KGs

Background



LLMs

- *Encoder* - unsupervised
  - Understanding entire sentence (text classification, NER)
- *Encoder-Decoder*
  - Training strategies in encoder-decoder LLMs can be more flexible (summary, translate, QA)
- *Decoder*
  - Only adopt the decoder module to generate target output text

KGs

- encyclopedic KGs
- commonsense KGs
- domain-specific KGs
- multi-modal KGs

Roadmap

Llm-Augmented KGs

- Existing methods in KGs fall short of handling incomplete KGs [33]
- Researchers take advantage of LLMs to process the textual corpus in the KGs and then use the representations of the text to enrich KGs representation [94]
- Recent studies try to design a KG prompt that can effectively convert structural KGs into a format that can be comprehended by LLMs.

> **Important**
>
> Some studies also use LLMs to process the original corpus and extract relations and entities for KG construction [95].

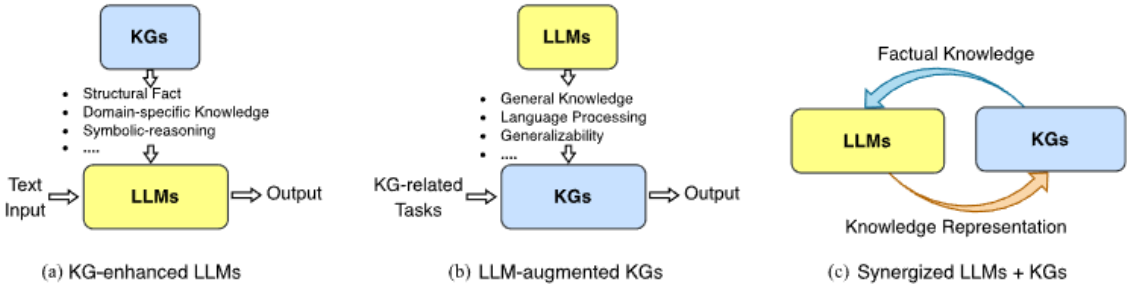
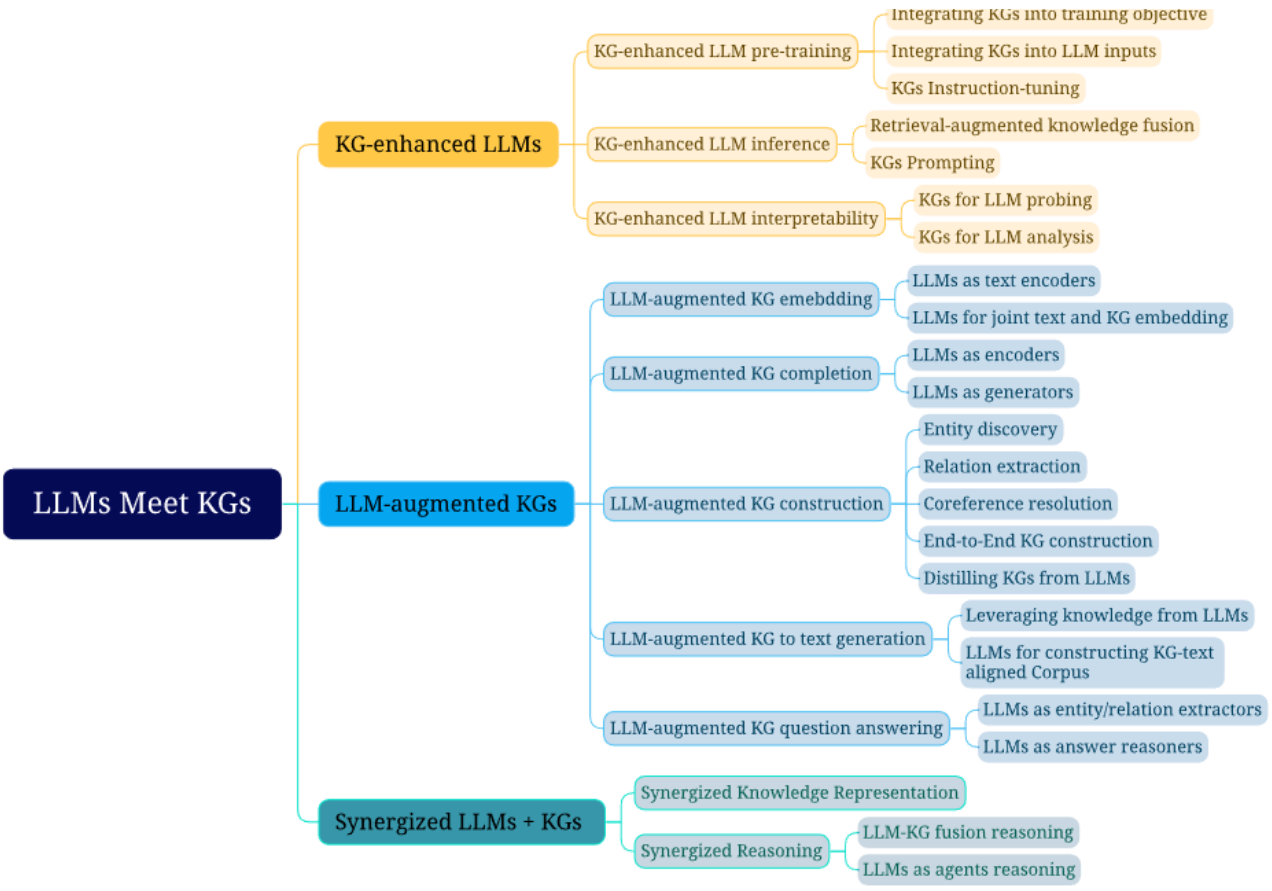


Fig. 6. General roadmap of unifying KGs and LLMs.

Categorization of unifying LLMs and KGs

- KG-enhanced LLMs
  - pre-training
  - inference
  - interpretability
- *LLM-augmented KGs*

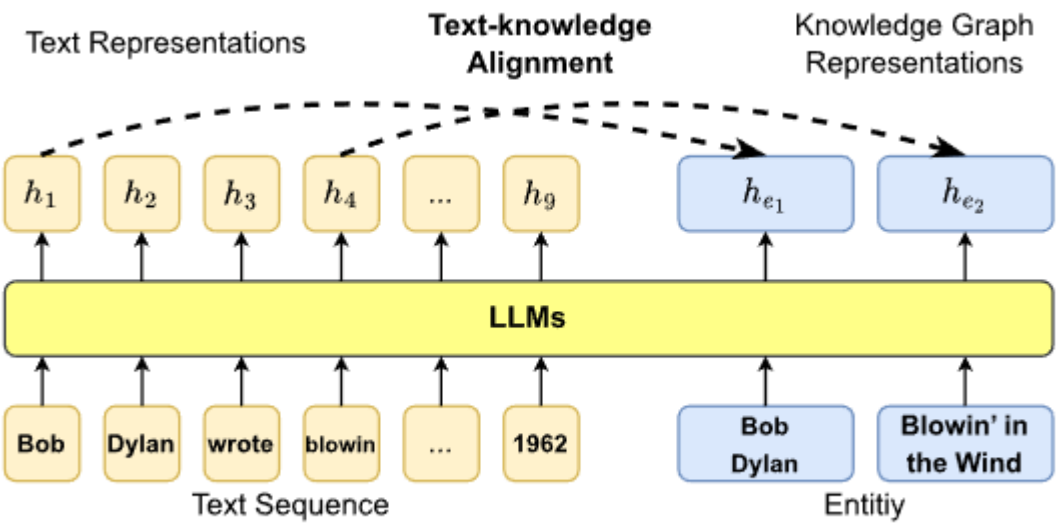
    - embedding
    - completion
    - construction
    - KG-to-text Generation
    - KG question answering
- Synergized LLMs + KGs
    - Synergized Knowledge Representation
    - Synergized Reasoning



KG-enhanced LLMs

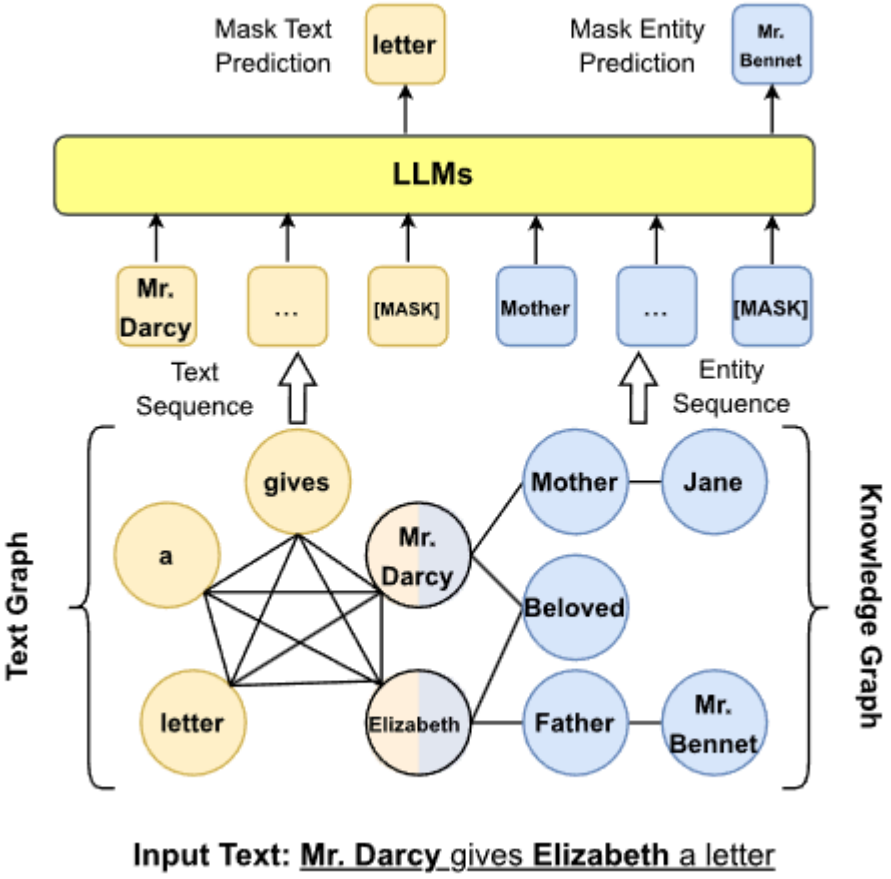
Pre-training

- *Integrating KGs into training objective*
    - intuitive idea is to expose more knowledge entities in the pre-training objective
    - entities that can be reached within a certain number of hops are considered to be the most important entities for learning, and they are given a higher masking prob- ability during pre-training
    - the other line of work explicitly leverages the connections with knowledge and input text
    - ERNIE feeds both sentences and corresponding entities mentioned in the text into LLMs, and then trains the LLMs to predict alignment links between textual tokens and entities in knowledge graphs
    - Deterministic LLM [104] focuses on pre-training language models to capture deterministic factual knowledge



Input Text: **Bob Dylan** wrote **Blowin' in the Wind** in 1962

- *Integrating KGs into LLM inputs*
  - **this kind of research focus on introducing relevant knowledge sub-graph into the inputs of LLMs**
  - knowledge graph triple and the corresponding sentences, ERNIE 3.0 [101] represents the triple as a sequence of tokens and directly concatenates them with the sentences
  - however, such direct knowledge triple concatenation method allows the tokens in the sentence to intensively interact with the tokens in the knowledge sub-graph, which could result in *Knowledge Noise* [36]
  - *to solve this issue*, K-BERT [36] takes the first step to inject the knowledge triple into the sentence via a visible matrix where only the knowledge entities have access to the knowledge triple information, while the tokens in the sentences can only see each other in the self-attention module
  - above methods can indeed inject a large amount of knowledge into LLMs. However, they mostly focus on popular entities and *overlook* the low-frequent and long-tail ones
  - DkLLM [108] aims to *improve the LLMs representations towards those entities*
  - Furthermore, Dict-BERT [125] proposes to leverage external dictionaries to solve this issue



- *KGs Instruction-tuning*
  - instead of injecting factual knowledge into LLMs, the KGs Instruction-tuning aims to *finetune LLMs* to better comprehend the structure of KGs and effectively follow user instructions to conduct complex tasks
  - utilizes both facts and the structure of KGs to create *instruction-tuning datasets*
  - KP-PLM [109] first designs several prompt templates to transfer structural graphs into natural language text. Then, two self-supervised tasks are proposed to finetune LLMs to further leverage the knowledge from these prompts.
  - OntoPrompt [110] proposes an ontology-enhanced prompt-tuning that can place knowledge of entities into the context of LLMs, which are further finetuned on several downstream tasks.

Inference

The above methods could effectively fuse knowledge into LLMs. However, real-world knowledge is subject to change and the limitation of these approaches is that they do not *permit updates to the incorporated knowledge without retraining the model*.

- *Retrieval-Augmented Knowledge Fusion*
  - The key idea is to retrieve relevant knowledge from a large corpus and then fuse the retrieved knowledge into LLMs
- *KGs Prompting*
  - To better feed the KG structure into the LLM during inference, KGs prompting aims to design a crafted prompt that converts structured KGs into text sequences, which can be fed as context into LLMs.

Comparison Between KG-Enhanced LLM Pre-Training and Inference

- pre-trained may have inference optimisation but it can't handle dynamic knowledge
- depends on application

Interpretability

Although LLMs have achieved remarkable success in many NLP tasks, they are still criticized for their lack of interpretability. The large language model (LLM) interpretability refers to the understanding and explanation of the inner workings and decision-making processes of a large language model [17].

- KGs for LLM Probing

Fig. 12. General framework of using knowledge graph for language model probing.

- KGs for LLM Analysis

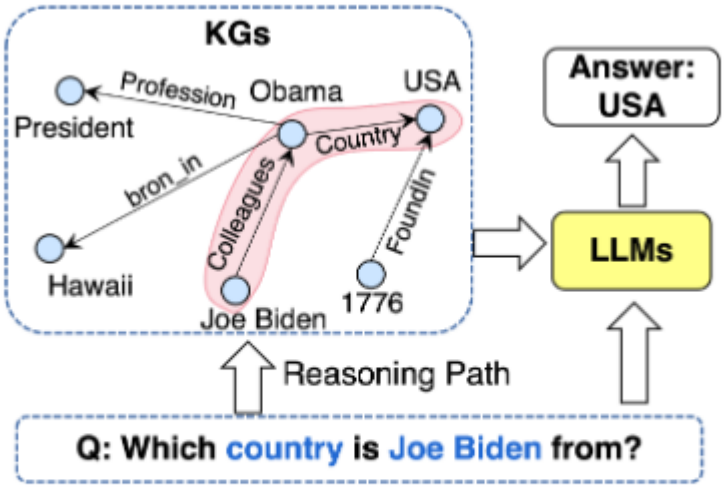


Fig. 13. General framework of using knowledge graph for language model analysis.

LLM-AUGMENTED KGS

LLM-Augmented KG Embedding

Knowledge graph embedding (KGE) aims to map each *entity and relation* into a *low-dimensional* vector (embedding) space.

- *LLMs as Text Encoders*
  - Conventional knowledge graph embedding methods mainly rely on the structural information of KGs
  - these approaches often fall short in representing unseen entities and long-tailed relations due to their limited structural connectivity [135], [136]
  - recent research adopt LLMs to enrich representations of KGs by encoding the textual descriptions of entities and relations [40], [94]

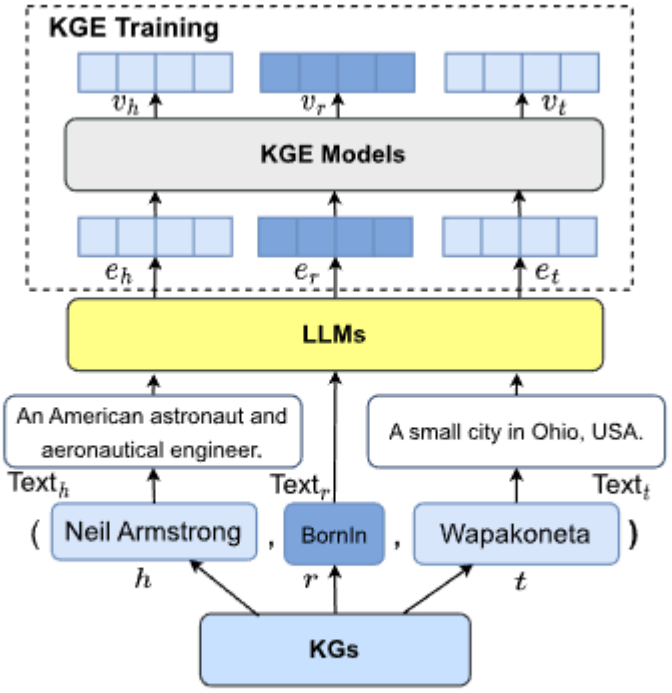


Fig. 14. LLMs as text encoder for knowledge graph embedding (KGE).

- *LLMs for Joint Text and KG Embedding*
  - Instead of using KGE model to consider graph structure, another line of methods directly employs LLMs to incorporate both the graph structure and textual information into the embedding space simultaneously [137], [138], [139].
  - they treat the entities and relations as special tokens in the LLM
  - During training, it transfers each triple and corresponding text description into a sentence where the tailed entities are replaced by [MASK]. The sentence is fed into a LLM, which then finetunes the model to predict the masked entity.

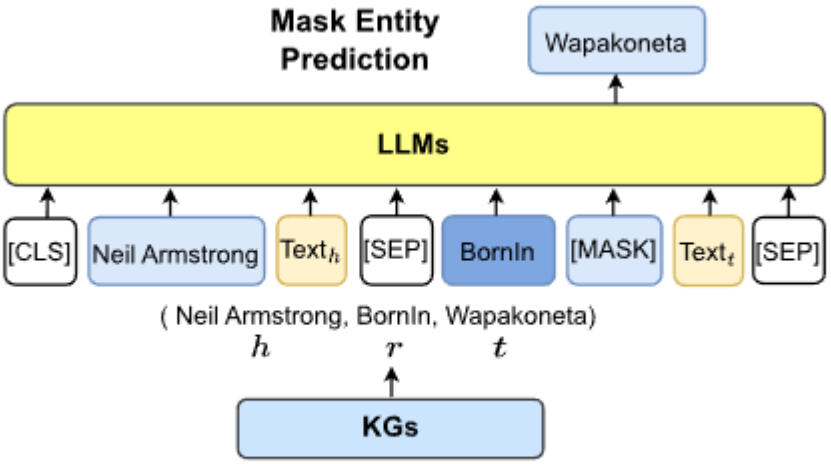


Fig. 15. LLMs for joint text and knowledge graph embedding.

LLM-Augmented KG Completion

Knowledge Graph Completion (KGC) refers to the task of *inferring missing facts* in a given knowledge graph. Similar to KGE, conventional KGC methods mainly focused on the structure of the KG, without *considering the extensive textual information*. However, the recent *integration of LLMs* enables KGC methods to encode text or generate facts for *better* KGC performance.

- *LLM as Encoders (PaE)*
  - first uses encoder-only LLMs to encode textual information as well as KG facts
  - then, they predict the plausibility of the triples or masked entities by feeding the encoded representation into a prediction head, which could be a simple MLP or conventional KG score function (e.g., TransE [33] and TransR [144])
- *LLM as Generators (PaG)*
  - *recent works* use LLMs as sequence-to-sequence generators in KGC [96], [145], [146]
  - involve encoder-decoder or decoder-only
  - LLMs receive a sequence text input of the query triple $(h, r, ?)$, and generate the text of tail entity t directly
  - For closed-source LLMs (e.g., ChatGPT and GPT-4), *AutoKG* adopts prompt engineering to design customized prompts [93]
  - these prompts contain the task description, few-shot examples, and test input, which instruct LLMs to predict the tail entity for KG completion

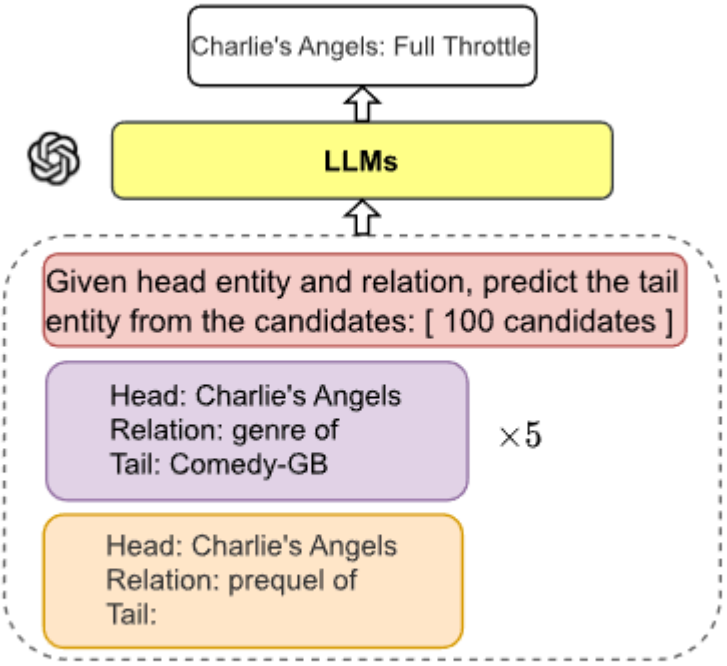


Fig. 16. Framework of prompt-based PaG for KG Completion.

Comparison between PaE and PaG

LLMs as Encoders (PaE) applies an additional prediction head on the top of the representation encoded by LLMs. Therefore, the *PaE framework is much easier to finetune* since we can only optimize the prediction heads and freeze the LLMs
However, during the inference stage, the PaE requires to compute a score for every candidate in KGs, which could be computationally expensive. Furthermore, the PaE requires the representation output of the LLMs, whereas some state-of-the-art LLMs (e.g. GPT-4 1) are closed sources and do not grant access to the representation output.

LLMs as Generators (PaG) does not need the prediction head, can be used without finetuning or access to representations. In addition, PaG directly generates the tail entity, making it efficient in inference without ranking all the candidates and easily generalizing to *unseen entities*. However, *generated entities could be diverse and not lie in KGs*. Also *slower for inference due to being auto-regressive*.

> **Important**
>
> Last, how to design a powerful prompt that feeds KGs into LLMs is still an open question.

Consequently, while PaG has demonstrated *promising* results for KGC tasks, the trade-off between *model complexity and computational efficiency* must be carefully considered when selecting an appropriate LLM-based KGC framework.

LLM-Augmented KG Construction

The process of knowledge graph construction typically involves multiple stages, including

1. *entity discovery* [147], [148], [149], [150]
2. *coreference resolution* [151], [152], [153]

3. *relation extraction* [154], [155], [156]
   Recent approaches have explored
4. *end-to-end knowledge graph construction*
5. *distilling knowledge graphs from LLMs*

- *End-to-End KG Construction*
  - *Kumar et al.* [95] propose a unified approach to build KGs from raw text, which contains two LLMs powered components
    - first finetune a LLM on named entity recognition tasks to make it capable of recognizing entities in raw text
    - then, they propose another “2-model BERT” for solving the relation extraction task, which contains two BERT-based classifiers
    - (no drawback mentioned)
  - *Guo et al.* [157] propose an end-to-end knowledge extraction model based on BERT, which can be applied to construct KGs from Classical Chinese text
  - *Grapher* [41] presents a novel end-to-end multi-stage system
    - first utilizes LLMs to generate KG entities, followed by a simple relation construction head, enabling efficient KG construction from the textual description
  - *PiVE* [158] proposes a prompting with an iterative verification framework that utilizes a smaller LLM like T5 to correct the errors in KGs generated by a larger LLM
- *Distilling Knowledge Graphs from LLMs*
  - *Some* research aims to distill knowledge from LLMs to construct KGs
  - *COMET* [159] proposes a commonsense transformer model that constructs commonsense KGs by using existing tuples as a seed set of knowledge on which to train
    - Using this seed set, a LLM learns to adapt its learned representations to knowledge generation, and produces novel tuples that are high quality
    - *Experimental results* reveal that *implicit knowledge from LLMs is transferred to generate explicit knowledge* in commonsense KGs

LLM-Augmented KG-to-Text Generation

The goal of Knowledge-graph-to-text (KG-to-text) generation is to generate high-quality texts that accurately and consistently describe the input knowledge graph information [160].

KG-to-text generation connects knowledge graphs and texts, significantly improving the applicability of KG in more realistic NLG scenarios, including storytelling [161] and knowledge- grounded dialogue [162].

However, it is challenging and costly to collect large amounts of graph-text parallel data, resulting in insufficient training and poor generation quality.

Two approaches *Leveraging Knowledge from LLMs* (represent KG in linear way and use LLM to construct text corpus from this) and *Constructing large weakly KG-text aligned Corpus*.

This doesn't seem to be relevant.

LLM-Augmented KG Question Answering

Knowledge graph question answering (KGQA) aims to find answers to *natural language questions* based on the structured facts stored in knowledge graphs [167], [168].

- *LLMs as Entity/relation Extractors*
  - Entity/relation extractors are designed to identify entities and relationships mentioned in natural language questions and retrieve related facts in KGs
- *LLMs as Answer Reasoners*
  - Answer reasoners are designed to reason over the retrieved facts and generate answers

Synergized LLMs + KGs

LLMs can be used to understand natural language, while KGs are treated as a knowledge base

- *Synergized Knowledge Representation*
  - the knowledge in text corpus is usually *implicit* and *unstructured*, while the knowledge in KGs is *explicit* and *structured*
  - synergized model can provide a better understanding of the knowledge from both sources, making it valuable for many downstream tasks
- *Synergized Reasoning*
  - To better utilize the knowledge from text corpus and knowl- edge graph reasoning, Synergized Reasoning aims to design a synergized model that can effectively conduct reasoning with both LLMs and KGs.
  - *LLM-KG Fusion Reasoning*
    - LLM-KG Fusion Reasoning leverages two separated LLM and KG encoders to process the text and relevant KG inputs [182]. These two encoders are equally important and jointly fusing the knowledge from two sources for reasoning
  - *LLMs as Agents Reasoning* <- what we wanted to do
    - Instead using two encoders to fuse the knowledge, LLMs can also be treated as agents to interact with the KGs to conduct reasoning [184], as illustrated in Fig. 21. KD-CoT [185] iteratively retrieves facts from KGs and produces faithful reasoning traces, which guide LLMs to generate answers.

Future directions

- the issue of *hallucination* may continue to persist in the realm of LLMs for the foreseeable future
  - further studies combine LLMs and KGs to achieve a generalized fact-checking model that can detect hallucinations across domains [189]
- some research efforts proposed for *editing* knowledge in LLMs [190] without re-training the whole LLMs
  - such solutions still suffer from poor performance or computational overhead
- it is impossible to follow *conventional KG injection* approaches described [38], [182] that change LLM structure by adding additional knowledge fusion modules on closed source models
  - converting various types of knowledge into different text prompts seems to be a feasible solution
  - However, it is unclear whether these prompts can generalize well to new LLMs
  - the prompt-based approach is limited to the length of input tokens of LLMs
- Multi-Modal LLMs for KGs
- LLMs for Understanding KG Structure
- Synergized LLMs and KGs for Birectional Reasoning