# Generating Domain-Specific Knowledge Graphs from Large Language Models

**Marinela Parović**[♣] **Ze Li**[♠] **Jinhua Du**[♣]

[♣]Huawei London Research Centre
[♠]Huawei Technologies Co., Ltd, Hangzhou, China
{marinela.parovic,lize23,jinhua.du}@huawei.com

## Abstract

Knowledge graphs (KGs) have been a cornerstone of search and recommendation due to their ability to store factual knowledge about any domain in a structured form enabling easy search and retrieval. Large language models (LLMs) have shown impressive world knowledge across different benchmarks and domains but their knowledge is inconveniently scattered across their billions of parameters. In this paper, we propose a prompt-based method to construct domain-specific KGs by extracting knowledge solely from LLMs' parameters. First, we use an LLM to create a schema for a specific domain, which contains a set of domain-representative entities and relations. After that, we use the schema to guide the LLM through an iterative data generation process equipped with Chain-of-Verification (CoVe) for increased data quality. Using this method, we construct KGs for two domains: books and landmarks, which we then evaluate against Wikidata, an open-source human-created KG. Our results show that LLMs can generate large domain-specific KGs containing tens of thousands of entities and relations. However, due to the increased hallucination rates as the procedure evolves, the utility of large-scale LLM-generated KGs in practical applications could remain limited.

## 1 Introduction

Large language models (LLMs) have emerged as a centerpiece of NLP. These models learn to understand and produce natural language(s) while also encoding a vast amount of world knowledge from the terabytes of text they "see" during their pre-training (Hoffmann et al., 2022; Touvron et al., 2023; DeepSeek-AI et al., 2024). Even early variants of pre-trained language models were found to contain rich factual knowledge (Petroni et al., 2019). These models (Devlin et al., 2019; Liu et al., 2020) contained millions of parameters and were trained on gigabytes of text, while the latest LLMs contain billions of parameters and are trained on terabytes of text containing resources such as scientific papers, Wikipedia, books, and web. As a consequence, they have been found to store a remarkable knowledge about many diverse domains (Brown et al., 2020; Jiang et al., 2021; Sun et al., 2024) and their abilities are commonly evaluated on open-domain question answering benchmarks such as TriviaQA (Joshi et al., 2017), NaturalQuestions (Kwiatkowski et al., 2019) and OpenBookQA (Mihaylov et al., 2018). The knowledge stored in LLMs is a by-product of their training procedure. However, this knowledge is encoded within their parameters which makes it difficult and inefficient to extract or use.

Knowledge graphs (KGs) store large-scale factual knowledge in a structured form, which enables an easy integration with query languages and search engines (Vrandečić and Krötzsch, 2014; Auer et al., 2007). They serve knowledge from different domains in the form of `entities` (nodes) and `relations` (edges) which form triples `[subject, relation, object]`. However, their construction and maintenance requires human efforts and large costs because KGs need to store comprehensive, up-to-date and correct knowledge (Noy et al., 2019). Consequently, methods for KG construction which automatically extract structured knowledge (in the form of triples) from unstructured texts have been developed (Lu et al., 2022; Chen et al., 2024; van Cauter and Yakovets, 2024; Zhang and Soh, 2024). In addition to general-domain KGs, such methods also target domain-specific KG construction using corpora such as medical records (Arsenyan et al., 2024) or maintenance texts (van Cauter and Yakovets, 2024). However, the applicability of these methods requires access to domain-specific documents which can be difficult or expensive to obtain for all domains of interest. To overcome this constraint and owing to the growing LLM memorisation and generation capa-



*Findings of the Association for Computational Linguistics: ACL 2025*, pages 11558–11574

bilities, Hu et al. (2024) proposed a prompt-based procedure to construct a general-domain KG from scratch relying solely on the knowledge encoded in LLMs parameters. This approach enables creating a large KG (millions of entities and relations), but their procedure does not offer any control over the obtained entity or relation types. This could be undesirable for targeted uses cases or applications where domain or schema are typically fixed (Abu-Salih, 2021). Cohen et al. (2023) proposed to construct a knowledge base centered around a given "seed" entity, but they perform only up to two hops resulting in high-precision KGs with only several dozen facts. Like Hu et al. (2024), they do not have a fixed set of entities and relation types, which offers little control over the generation process and makes the evaluation more challenging.

In this paper, we study the feasibility of constructing large, domain-specific KGs from scratch using solely LLMs. To this end, we propose a prompt-based method for extracting domain-specific knowledge that model has seen during pre-training. More precisely, given a domain of interest, we first generate a set of entities and relations (a schema) which contains relevant properties describing the domain. These entities and relations are then used during an iterative, prompt-based data generation procedure to guide the model towards producing an "exhaustive" KG about the given domain. Finally, the quality of the constructed KG is evaluated against ground-truth human-crafted KGs. Our method provides a way of representing the LLM knowledge about any domain in a structured form. It does so by offering a schema-guided procedure which converts the knowledge stored within LLMs' billions of parameters into a domain-specific graph with arbitrary number of hops, which can serve different applications.

In summary, our main contributions are: 1) we formulate an iterative, prompt-based algorithm for extracting domain-specific KGs directly from LLMs (LLM is used only for inference); 2) we propose to systematically evaluate the quality of obtained KGs against existing human-created KGs; 3) our method goes beyond KG construction by providing insights into domain knowledge stored within LLMs, and can serve for evaluation or comparison between LLMs. Our code and KGs are publicly available at `http://github.com/parovicm/llm-kg-gen`.

## 2 Methodology

### 2.1 Background

The process of KG construction traditionally relies on human experts to define the structure of domain knowledge by establishing relevant entities and relations (so-called schemas which is the term used throughout this paper[1]), which are then populated with data (in the form of triples which are pairs of entities connected by relations or pairs of KG nodes with an edge between them). This process is known to be lengthy and expensive requiring access to large amounts of domain-specific documents which are mined by human experts, rule-based methods or NLP systems (Noy et al., 2019; Zhang and Soh, 2024). Recently, Hu et al. (2024) and Cohen et al. (2023) proposed to construct KGs by extracting data directly from LLMs (no external documents). Hu et al. (2024) construct a large, general-domain KG, but their procedure offers no control over the domain, entities or relations of the obtained KGs. Cohen et al. (2023) expand a KG given a "seed" entity but only create graphs with several dozens of facts. Unlike them, we propose a method to create large domain-specific KGs. Given only a domain name, we first generate a KG schema. Using this schema, we iteratively guide the LLM to generate data according to the schema content, while relying on the model's capability to store and retrieve rich knowledge about different domains.

### 2.2 Schema Generation

The schema of a knowledge graph consists of entities, attributes, and relations. Entities represent main concepts present in the domain, attributes are various properties which further describe entities, and relations connect pairs of entities. In order to construct a schema for a given domain, we assume access to a single human-crafted KG schema. This ensures that created schemas adhere to a predefined structure. Given a name of the domain of interest, we employ a two-step prompt-based approach to generate a schema. In the first step, we generate a set of entities and relations, while the second step adds attributes to entities. An example of a simplified schema for KG about *books* is given in Figure 1, with the full schema available in Appendix B.

[1]The term ontology is also prevalent but it more commonly used to describe larger and broader hierarchies of knowledge.

```
{
    "concepts": {
        "book": {
            "attributes": {
                "title": "string",
                "publication_date": "string",
                "ISBN": "string"
            }
        },
        "author": {
            "attributes": {
                "author_name": "string",
                "nationality": "string",
                "birth_date": "string"
            }
        },
    },
    "relations": {
        "author_writes_book": {
            "source": "author",
            "target": "book",
            "name": "writes"
        },
    }
}
```


Figure 1: Schema for *books* domain, containing entities book and author with attributes such as ISBN of a book or author's date of birth, and a relation between them (author_writes_book).

### 2.3 Data Generation

We formulate an iterative algorithm for domain-specific KG construction. Using generated schema, we guide the LLM to generate entities, relations and attributes. For simplicity, our algorithm assumes there is one "main" entity in the set of entities which appears in all relations.[2] We also assume that all entities have one "main" attribute which identifies them (for example, book's title or author's name) and we call this attribute *identifier*. Our algorithm consists of the following three steps:

1. **Initial Relation Generation**: for all relations in the schema, generate $N$ (initial) pairs of entity identifiers which satisfy these relations;
2. **Entity Completion** (EnComp): given a pair of entity identifiers, complete all the remaining entities and attributes in the schema;
3. **Entity Generation** (EnGen): given two entity identifiers satisfying one of the relations (one of them is the "main" entity), generate other $M$ main entity identifiers satisfying that relation.

The role of the step 1. is to generate initial data which will act as a starting point for further generation. Typically, this step generates well-known, popular data, which has a large presence in the model's pretraining corpora. The role of step EnComp is to complete the schema for generated pairs of identifiers by generating values for remaining entities and attributes. For example, given a book's title and author's name (these are *identifiers*), this step will generate the remaining entities in the book schema and attributes for book, author and other entities. Lastly, step EnGen is crucial for increasing the size of KG since it generates a list of main entity's identifiers which satisfy certain condition. An example of this is given book's title and author's name, this step asks the model to generate other $M$ book titles written by a given author. In principle, steps EnComp and EnGen are interchangable and we apply them repeatedly to increase the size of KG.[3] All prompts are given in Appendix D.

During a single step, we mark "newly" generated pairs of identifiers, and use them for prompting LLM to generate more data in the subsequent step. For example, step EnComp is used to "complete" newly generated pairs of identifiers which have not been completed previously. Similarly, step EnGen will be used to generate identifiers for pairs which have not already been used. Therefore, the LLM in never prompted twice in the same way.

### 2.4 Data Validation

As the generation process progresses, the model's ability to generate factually correct data decreases and the quality of the generated data drops. This is expected due to growing model's uncertainty on long-tail data (Zheng et al., 2024; Hu et al., 2024). While step 1. generates factual data, further steps introduce hallucinations. Unsurprisingly, we observe that step EnGen, whose role is to scale up the KG size, is the most prone to hallucinations. More precisely, there is a trade-off between increasing the size of KG with larger values of $M$ (the number of new identifiers generated) and the amount of non-factual knowledge introduced in this step. Put simply, one can attempt to scale up the KG by increasing value $M$, but this will decrease the KG quality.

To address the issue of hallucinations, and increase the factuality of generated data, we equip step EnGen with a recent approach Chain-of-Verification (CoVe) (Dhuliawala et al., 2024). This approach is used for mitigating hallucinations by

[2] For example, in the *books* domain, book is the main entity and it has relations with all other entities (author, publisher etc). While a relation between author and publisher would be valid, we do not consider such relations. Nonetheless, author and publisher will still be indirectly connected via book entity.

[3] While we apply step EnComp first, the order of steps EnGen and EnComp can be altered and it does not significantly impact the final KG.

relying exclusively on the knowledge that LLM holds, which satisfies our setup. CoVe operates in the following steps:

1. Generate an initial response to a given query (a list of main entity identifiers satisfying given prompt).
2. Given a query, generate templates for verification questions based on an example of query and templates.
3. Generate a list of verification questions to fact-check the initial response. LLM generates questions based on templates and there are typically 1-2 questions for each generated identifier.
4. Answer each generated question independently to minimise the effect of answers on each other.
5. Output an updated response by using query, initial response and verification questions and answers.

Prompts for all steps are given in Appendix E.

## 3 Experimental Setup

### 3.1 LLMs

We conduct our experiments using two open-source state-of-the-art (SotA) LLMs with ~70B parameters: `Llama3.1-70b` (Dubey et al., 2024) and `Qwen2.5-72b` (Yang et al., 2024). In the rest of the paper we denote them simply as `Llama` and `Qwen`. Due to computational requirements of our experiments and the model size, we employ quantised versions of both models with Int4 precision for faster inference. We host models using `vllm` since our method makes no modifications to models and requires only inference.[4] We fix the temperature parameter to 0.2 throughout both schema and data generation process. This value worked well in our preliminary experiments and low temperature supports maximising data factuality. To enable the reproducibity of our results, we rely exclusively on open-source models but our method can be used with any model available for inference.[5]

### 3.2 Domains

We evaluate our method on two different domains: *books* and *landmarks*.[6] Schemas are generated with a prompt-based approach (§2.2) using `Llama` model. Schemas for both domains and prompts for generating them are given in Appendix B and C, respectively.

### 3.3 KG Generation

In **Initial Relation Generation** step of our algorithm (§2.3), we set value $N$ to 20 because this tends to produce correct pairs of identifiers. Repetitions or hallucinations appear if $N$ is set to higher values such as 50 or 100. These issues are relation-dependent and more profound for relations: 1) which are harder to generate (LLMs have likely seen less data about them), 2) where one of the identifiers tends to have missing values, or 3) which have less than $N$ real values. For example, parameter $N$ could be set to a higher value when generating identifiers for the relation `author_writes_book` than for relations `book_belongs_to_series` or `book_features_character`, because the model is better at generating entity `author` than entity `character`, and because the majority of `books` do not belong to a `series`. For simplicity, we keep value $N = 20$ for all relations to optimise generality and data quality in this step.

Following step 1, we alternate between steps `EnComp` and `EnGen` (§2.3). In total, we apply `EnComp` four times and `EnGen` three times. While more iterations would lead to a larger amount of data, they would also increase the number of hallucinations and generation time. Value $M$ in step `EnGen` determines the trade-off between hallucinations and data quantity. Similarly to step 1, suitable values for $M$ are relation-dependent or example-dependent since some `authors` wrote only a single `book` while others wrote dozens. Similarly, most `characters` appear only in a single `book`. Additionally, we expect there are many `landmarks` in a given `country`, or with a given `style`, but only a few designed by a specific `architect`. To keep our method domain- and relation-independent we fix value $M$ to 10. However, we realise that better performance could be achieved by carefully tuning $M$ based on specific entity or relation types, or their values. For example, Hu et al. (2024) propose to first ask the LLM to output the number of examples it believes to know, followed by asking it to generate predicted number of examples.

[4] https://github.com/vllm-project/vllm

[5] Models hidden behind APIs can be updated or entirely removed, and the cost from using them could become too high.

[6] These domains are different in nature. Landmarks include famous real buildings or geographical objects. We chose them because LLMs are likely to have a good knowledge about them, they are present in Wikidata, and they can be of interest for applications such as recommendation or advertisement.

In all data generation steps, we found it helpful to encourage the model to output None if it cannot provide an answer, which has been found effective previously (Alivanistos et al., 2022; Hu et al., 2024; Cohen et al., 2023).

| Generated Entity | Wikidata Item or Property |
|---|---|
| book | written work (Q47461344) |
| author | author (P50) |
| publisher | publisher (P123) |
| genre | genre (P136) |
| award | award received (P166) |
| character | characters (P674) |
| series | part of the series (P179) |

Table 1: Wikidata categories used for evaluating KG about *books*.

| Entity or Relation | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| book | 16,373 | 35,173 | 17,006 | 55,225 |
| author | 3,986 | 8,539 | 5,583 | 12,848 |
| publisher | 1,337 | 1,857 | 1,711 | 3,341 |
| genre | 907 | 967 | 760 | 1,466 |
| award | 118 | 195 | 448 | 538 |
| character | 1,851 | 3,629 | 5,586 | 7,588 |
| series | 1,894 | 2,689 | 2,495 | 6,044 |
| Total | **26,466** | **53,049** | **33,589** | **87,050** |
| author_book | 15,855 | 35,371 | 18,082 | 58,302 |
| book_publisher | 13,485 | 28,304 | 18,700 | 58,992 |
| book_genre | 17,296 | 35,897 | 19,684 | 62,346 |
| book_award | 1,727 | 4,748 | 4,205 | 8,994 |
| book_character | 4,576 | 11,125 | 12,041 | 32,380 |
| book_series | 5,857 | 10,779 | 7,562 | 21,475 |
| Total | **58,796** | **126,224** | **80,274** | **242,489** |

(a) Domain: *books*

| Entity or Relation | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| landmark | 16,396 | 35,994 | 30,349 | 49,738 |
| city | 1,924 | 2,871 | 4,328 | 5,735 |
| country | 135 | 168 | 243 | 436 |
| architect | 1,421 | 2,238 | 3,764 | 5,189 |
| style | 831 | 906 | 1,925 | 2,400 |
| event | 346 | 1,415 | 14,498 | 21,547 |
| organization | 1,172 | 2,324 | 9,983 | 14,902 |
| Total | **22,225** | **45,916** | **65,090** | **99,947** |
| landmark_city | 15,389 | 33,490 | 31,922 | 52,434 |
| landmark_country | 15,605 | 33,111 | 30,570 | 50,041 |
| landmark_architect | 4,978 | 14,522 | 13,075 | 22,606 |
| landmark_style | 10,668 | 21,426 | 30,733 | 50,184 |
| landmark_event | 1,756 | 8,766 | 22,795 | 39,606 |
| landmark_organization | 4,858 | 11,276 | 24,377 | 39,682 |
| Total | **53,254** | **122,591** | **153,472** | **254,553** |

(b) Domain: *landmarks*

Table 2: Number of generated entities and relations for *books* and *landmarks* with Llama and Qwen (with and without CoVe). Relations are denoted using pairs of entities (full relations names are given in Appendix B).

### 3.4 Data Evaluation

To evaluate generated KGs we rely on Wikidata, an open-source collaboratively edited KG (Vrandečić and Krötzsch, 2014). As of January 2025, Wikidata has 1.65 billion triples.[7]

| Entity | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| book | 71 | 996 | 4,451 | 16,373 |
| author | 66 | 364 | 1,302 | 3,986 |
| publisher | 61 | 229 | 579 | 1,337 |
| genre | 35 | 130 | 231 | 907 |
| award | 16 | 36 | 81 | 118 |
| character | 33 | 178 | 964 | 1,851 |
| series | 27 | 143 | 512 | 1,894 |
| Total | **309** | **2,076** | **8,120** | **26,466** |

(a) Domain: *books*

| Entity | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| landmark | 39 | 722 | 4,316 | 16,396 |
| city | 24 | 212 | 740 | 1,924 |
| country | 21 | 45 | 87 | 135 |
| architect | 26 | 189 | 581 | 1,421 |
| style | 24 | 105 | 323 | 831 |
| event | 15 | 79 | 152 | 346 |
| organization | 13 | 104 | 395 | 1,172 |
| Total | **162** | **1,456** | **6,594** | **22,225** |

(b) Domain: *landmarks*

Table 3: Number of generated entities with Llama after each step EnComp is finished (step EnComp is applied four times and columns correspond to them).

Given main entity for a domain, we first identify a Wikidata category (item) which corresponds to it. We then find relations (properties) which connect that main entity with all other entities in our schema. The list of Wikidata categories for the *books* domain is given in Table 1 and for the *landmarks* domain in Table 16. We first query Wikidata to obtain all data for the main entity. Using that data, we then obtain values of all other entities and relations (using properties) (see Table 1). This data is then used to evaluate generated entities and relations.[8]

**Entity Evaluation.** We compute the percentage of generated entities which appear among Wikidata entities using the exact match. More precisely, if the generated entity's identifier appears among the Wikidata entity names, that entity is considered correct. When applied directly, this metric is too strict and it discards many real entities due to punctuation, different formatting or paraphrasing. In order to account for variability of naming and weaken the strictness of exact match, we obtain aliases (alternative names, acronyms or abbreviations) for entities from Wikidata. For example, book *Harry Potter and the Philosopher's Stone*

[7] https://en.wikipedia.org/wiki/Wikidata

[8] Most of generated attributes (associated with entities) can be found as properties on Wikidata. For example, literary works have properties P1104 and P577 for number of pages and publication date. Both of these values are book's attributes in our schema. Therefore, they could also be extracted and used to evaluate entities or perform entity resolution.

| Entity | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| book | **54.0** | 41.0 | 44.1 | 25.7 |
| author | **69.3** | 58.2 | 61.3 | 49.8 |
| publisher | **52.0** | 50.3 | 41.4 | 31.3 |
| genre | 41.4 | **48.4** | 43.9 | 36.5 |
| award | **56.8** | 48.7 | 33.2 | 29.2 |
| character | **33.0** | 27.8 | 22.2 | 21.2 |
| series | **25.2** | 23.5 | 17.4 | 9.4 |
| Total | **52.4** | 42.5 | 41.1 | 28.1 |

(a) Domain: *books*

| Entity | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| landmark | **36.1** | 31.1 | 29.5 | 25.9 |
| city | **91.6** | 91.3 | 76.4 | 75.0 |
| country | **95.6** | 92.3 | 62.7 | 40.6 |
| architect | **59.2** | 55.9 | 37.2 | 34.0 |
| style | 29.7 | **31.1** | 12.3 | 11.2 |
| organization | **45.5** | 43.3 | 15.8 | 14.1 |
| Total | **43.2** | 37.1 | 30.9 | 27.4 |

(b) Domain: *landmarks*

Table 4: Overlap between generated entities and Wikidata entities for *books* and *landmarks* with Llama and Qwen (with and without CoVe). We exclude entity event from *landmarks* because it does not have a corresponding Wikidata entity.

| Relation | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| author_book | **86.5** | 76.0 | 79.6 | 61.5 |
| book_publisher | **39.1** | 31.6 | 25.7 | 17.7 |
| book_genre | **39.5** | 36.9 | 36.0 | 30.2 |
| book_award | **41.2** | 34.2 | 39.8 | 29.1 |
| book_character | **76.8** | 61.2 | 60.6 | 46.0 |
| book_series | **57.3** | 49.0 | 46.6 | 38.1 |
| Total | **59.2** | 52.2 | 50.0 | 39.4 |

(a) Domain: *books*

| Relation | Llama | | Qwen | |
|---|---|---|---|---|
| | CoVe | NO-CoVe | CoVe | NO-CoVe |
| landmark_city | **56.5** | 49.5 | 45.6 | 39.3 |
| landmark_country | **91.2** | 89.7 | 85.2 | 83.2 |
| landmark_architect | **59.7** | 35.6 | 37.1 | 29.8 |
| landmark_style | **23.9** | 17.0 | 13.3 | 11.0 |
| landmark_organization | **48.2** | 31.4 | 25.6 | 21.7 |
| Total | **63.1** | 54.8 | 49.8 | 45.3 |

(b) Domain: *landmarks*

Table 5: Overlap between generated relations and Wikidata relations for *books* and *landmarks* with Llama and Qwen (with and without CoVe).

has aliases *Harry Potter and the Sorcerer's Stone*, *first Harry Potter book*, and *Sorcerer's Stone*, while author *J. R. R. Tolkien* has aliases *J-R-R Tolkien*, *John Ronald Reuel Tolkien*, *John Tolkien*, *J.R.R Tolkien*, *John R. R. Tolkien*. The task of identifying different representations of the same entities is entity resolution and many methods have been developed for it (Kasai et al., 2019; Li et al., 2020; Wadhwa et al., 2024). The main focus of this work is to assess the feasibility of constructing large, accurate, domain-specific KGs from LLMs and we address entity resolution by relying on Wikidata aliases. However, future work could use entity resolution methods for evaluation to increase matching between our KGs and Wikidata.

**Relation Evaluation.** Once we match the main entities with Wikidata entities, we can evaluate the correctness of generated relations. For example, if we generated a book *Harry Potter and the Philosopher's Stone*, with author *J. K. Rowling*, this matches the author obtained from Wikidata making the relation correct. We report the percentage of correct relations among those where main entities exist in Wikidata.

## 4 Results and Discussion

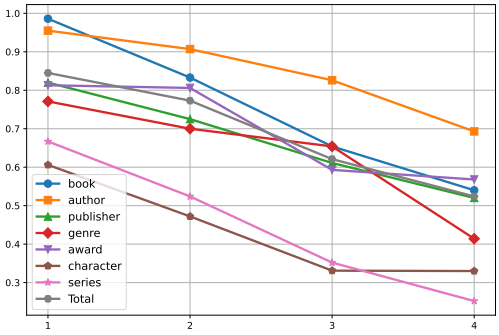
### 4.1 Data Quantity

Using our method, we generate KGs about *books* and *landmarks* using Llama and Qwen models. Step EnGen of our method employs CoVe (§2.4). As an ablation, we also report the results obtained without CoVe, denoted with -NO-CoVe. Throughout this section, we use terms quality and overlap to denote precision of our KGs, i.e. the proportion of generated data which is present in Wikidata.

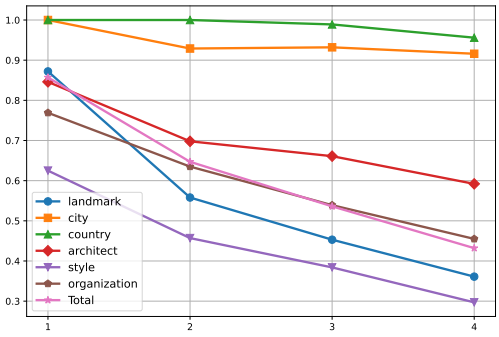
We report the unique number of entities and relations in our KGs in Table 2. For *books* domain, we obtain 26,466 entities and 58,796 relations with Llama, while Qwen produced 33,589 entities and 80,274 relations. For *landmarks* domain, we obtain 22,225 entities and 53,254 relations with Llama, and 65,090 entities and 153,472 relations with Qwen.

We make several observations: 1) Llama produces the same amount of data for both domains. This holds for both Llama and Llama-NO-CoVe versions; 2) Qwen produces much more data about *landmarks* than *books* (~2× more data for *landmarks* when CoVe is used); 3) Qwen generates significantly more data than Llama. The difference is much higher for *landmarks* where Qwen produced 40k more entities and 100k more relations than Llama when CoVe is used. The gap is even greater without CoVe. However, we stress that larger amount of data is valuable only when its quality is maintained.

**KG Size Change.** We monitor the change in the number of entities after every step EnComp is completed and observe different growth rates for different entities (Table 3). For example, book, author or character increase ~4× between steps EnComp




(a) Domain: *books*




(b) Domain: *landmarks*

Figure 2: Entity overlaps decrease during KG construction for all entities and both domains. We show values of Llama, while the remaining configurations are given in the Appendix.

2 and 3, while genre, award or publisher see more moderate increases of ~2×.

### 4.2 Entity Evaluation

**Main Observations.** Our main results from entity evaluation on both domains are given in Table 4. We observe that Llama achieves 10-14% higher performance than Qwen. Interestingly, Llama-NO-CoVe performs better than Qwen equipped with CoVe on both domains.

**Effect of CoVe.** We find that using CoVe leads to large performance gains. On *books* domain, using CoVe leads to average gains of 10-13% depending on the model, with more moderate gains of 3-6% on *landmarks* domain. While both models produce 2-3× more data when CoVe is not used (Table 2), the utility of the additional data might be limited due to its lower quality. However, CoVe makes the data generation process several times slower and it is by far the slowest step in our generation process.

This result suggests that using appropriate verification methods could prove critical in producing high quality KGs. Therefore, developing verification methods that have a better efficiency-accuracy trade-offs is a crucial step towards increasing the practical utilities of KGs obtained by extracting entities and relations directly from LLMs.

**Per-Entity Performance.** Our results demonstrate that LLMs' display large variance in their ability to generate different entities. For example, in the *books* domain, both models are much more successful when generating authors or genre (41.4-69.3%) than character or series (17.4-33%). Similarly, on the landmarks domain we observe that models (especially Qwen) struggle to generate style and organization for a landmark (12.3-45.5%), while being much more successful when generating cities or countries (62.7-95.6%). This is likely influenced by their pre-training data and training procedure.

**Size vs Quality.** Lastly, we report the drop in KG quality during generation process (the change in size is given in Table 3). In particular, Figure 2 shows how the overlap between Wikidata and generated data changes after each EnComp step. After the first EnComp, values for most entities are above 80%, and entities like book, author, city or country have precisions above 90%. However, at that point, only a few hundred of well-known entities have been generated. This observation is in agreement with Cohen et al. (2023), who show that it is possible to construct small KGs (~100 triples) around a given seed entity with high precision (even though they evaluate triples rather than entities). As the KGs grow, the overlap for all entities decreases and hallucinations increase, showing that extracting long-tail knowledge from LLMs remains challenging (Hu et al., 2024; Sun et al., 2024; Kandpal et al., 2023).

### 4.3 Relation Evaluation

Finally, in Table 5, we report results from evaluating generated relations. We again observe that both Llama and Llama-NO-CoVe outperform Qwen, and that certain relation exhibit much better scores than others.

### 4.4 Qualitative Analysis

The exact match used in our evaluation is a strict metric, even in the presence of Wikidata aliases. In addition, not all data generated by LLMs is guaranteed to be present in Wikidata. In order to gain further insight into the effectiveness of our evaluation procedure, we manually assess a small amount of generated data. In particular, we look at un-

| Entity Type | Main Entity | Entity | Wikidata Entity |
|---|---|---|---|
| author | the cather in the rye | j.d. salinger | jerome salinger, j. d. salinger, jerome david salinger |
| author | scaramouche | raphael sabatini | rafael sabatini |
| author | the high road | terry fallis | josephine edna o'brien, edna o'brien |
| author | fracture | megan miranda | jeff bond |
| publisher | the handmaid's tale | mcclelland and stewart | mcclelland & stewart, mcclelland & goodchild |
| publisher | the picture of dorian gray | ward, lock and company | lippincott's magazine, lippincott's monthly magazine |
| publisher | the age of innocence | d. appleton & company | aurum press |
| genre | the amazing adventures of kavalier & clay | historical fiction | historical novella, historical novel, superhero fiction |
| genre | the scarlet letter | romance | historic fiction, romantic fiction, love story, romance fiction, love fiction |
| genre | watchmen | graphic novel | alternate history comics, fantasy, fantasy fiction, superhero comic, superhero comics |
| award | to kill a mockingbird | pulitzer prize | pulitzer prize for fiction |
| award | king of morning, queen of day | philip k. dick award | premi robert holdstock a la millor novel·la fantàstica |
| award | clara callan | governor general's award | governor general's award for english-language fiction |
| character | the lord of the rings | frodo baggins | list of middle-earth characters |
| character | god emperor of dune | leto atreides ii | leto ii atreides, leto iii, duncan idaho |
| character | frankenstein; or, the modern prometheus | victor frankenstein | elizabeth frankenstein, elizabeth lavenza, doctor waldman, frankenstein's monster |
| series | the blade itself | the first law trilogy | the first law |
| series | the adventures of sherlock holmes | sherlock holmes | sherlock holmes novels |
| series | dune | dune series | dune, dune chronicles, list of games based on dune |

Table 6: Examples of correct relations from generated KG about *books* that are unmatched with Wikidata.

matched relations from *books* domain generated with Llama during the last ($4^{th}$) EnComp step. The statistics of this data are given in Table 3. We randomly select 30 samples from each relation in *books* domain, resulting in 180 evaluated examples. The evaluation reveals three types of errors: 1) the generated relation is wrong; 2) the generated relation is correct but unmatched due to different formatting or phrasing; 3) the generated relation is correct but missing in Wikidata. In Table 6, we show examples of error types 2) and 3) for different relations. Therefore, all these examples are correct but unmatched due to overly strict comparison using exact match or missing data in Wikidata. More precisely, among the evaluated 180 triples, we found that 77 are wrong (error type 1), 65 have error type 2) and the remaining 38 triples have error type 3). Therefore, the scores reported in Table 5 underestimate the actual quality of relations present in generated KGs.

Qualitative analysis of a subset of generated data reveals that the evaluation of (LLM-generated) KGs remains a challenge and that we undervalue data quality. As a conclusion, future work should employ more sophisticated entity resolution methods, such as those mentioned in Section 3.4 and develop entity- and relation-specific matching rules to overcome some of the observed issues.

## 5 Related Work

**KGs and LLMs.** Hao et al. (2023) propose to construct commonsense KGs from a given set of relations (for example, from ConceptNet (Speer et al., 2017)). Given a relation, the model is asked to generate pairs of entities which satisfy it. Our framework does not require any input entities or relations - they are generated by the LLM given a domain name. Cohen et al. (2023) propose to expand a KG around given entity, allowing the LLM to discover relation types instead. Their KGs are created by performing only two hops. In contrast, our framework has a fixed set of entities and relations which facilities easier evaluation and our KGs have many hops, containing tens of thousands of entities and relations.

Several works attempted to evaluate LLMs' capabilitites by formulating questions or introducing benchmarks based on open-source KGs (factual or commonsense) (Mallen et al., 2023; Luo et al., 2023; Huang et al., 2024; Liu et al., 2024).

**Factuality and Hallucinations.** Petroni et al. (2019) present an analysis of factual and commonsense knowledge present in BERT by formulat-

ing fill-in-the-blank cloze statements. Sun et al. (2024) create a benchmark Head-to-Tail to evaluate the factual knowledge of LLMs for entities with different levels of popularity. Their results show that LLMs have limited knowledge about requested entities and perform poorly on tail entities. Manakul et al. (2023) introduce a method to fact-check LLMs' responses, by sampling multiple response from the model and measuring the consistency between them. Their hypothesis is that when the model knows the concept well, these responses will be similar and consistent. Du et al. (2024) propose an approach motivated by the society of mind where multiple models propose and jointly debate their responses and reasoning processes to arrive to a common answer. Varshney et al. (2023) use model's uncertainty to detect false concepts in LLMs' responses. They then propose to repair these mistakes using retrieval-augmented LLMs. Dhuliawala et al. (2024) introduce CoVe, a method where model verifies its responses and self-corrects them. We equip our KG generation with this method. Li et al. (2024) and Zhang et al. (2023) contribute comprehensive surveys on different hallucination detection and mitigation techniques.

## 6 Conclusion

KGs have gained popularity in different applications due to their ability to store factual knowledge in a structured format, enabling efficient and reliable information search and retrieval. LLMs have achieved remarkable performance on many benchmarks recently and have been shown to store rich factual knowledge. However, this knowledge cannot be efficiently or conveniently used because it is encoded with their billions of parameters. In this paper, we proposed a procedure to create domain-specific KGs by extracting knowledge from LLMs' parameters, requiring only a domain name as input. Our procedure includes schema generation, data generation and verification. In this way, we generated KGs about books and landmarks containing tens of thousands of entities and relations and evaluated their quality against `Wikidata`. We also showed that `Llama` significantly outperforms `Qwen` on this task, producing higher-precision KGs. Our method goes beyond KG construction and can be used for evaluating or comparing different LLMs.

## Limitations

The bottleneck in our framework is the Chain-of-Verification method, which is by far the most time-consuming step. For example, using `Llama`, one `EnComp` prompt takes ~8s, an `EnGen` prompt takes ~2.5s on average and a single CoVe step (which involves several prompts) takes ~40s on average. Although finding methods that offer a more favorable trade-off between accuracy and efficiency would significantly improve our method, this exploration goes beyond the scope of this paper.

**Validity of Findings.** Due to computational requirements of KG construction, the evaluation of this paper is conducted on two domains and with two LLMs which poses concerns about generalisability of our findings across domains and LLMs. However, using two distinct domains coupled with two state-of-the-art open-source LLMs adds confidence our findings are correct.

**Prompt Sensitivity.** Our method is based on prompting which makes it sensitive to prompt formulations and some prompt tuning might be required if new models are used. However, our prompts are short and simple, and thus the efforts of prompt engineering for new models would be minimal.

**Open-Source Models.** Our evaluation contains only open-source models. We choose not to experiment with models like `ChatGPT` and `GPT4` due to large cost this would incur as well as harder reproducibility in case these models are updated or no longer available. However, the quality of these models is unquestionable and they would likely perform well on KG construction task too.

**Risks.** This procedure can be used to generate harmful or toxic data, where users may replace domain, schema or model with those that produce harmful values. We also acknowledge that LLMs can give biased results due to their training data and training procedures.

**Non-Factuality.** The data generated by our procedure contains hallucinations, i.e. it is not factual and should not be used in places where factuality is necessary. LLMs are known for their hallucination issues and many methods have been proposed to try and mitigate them. One such method is Chain-of-Verification used in this work, which reduces the hallucinations but it does not completely remove them.

## Acknowledgments

We would like to thank Jeff Pan and his team from Huawei Edinburgh Research Centre for sharing their work on large language models for knowledge graph construction. We are also grateful to Yiqin Li for his work on Chinese knowledge graph construction during an internship with us.

## References

Bilal Abu-Salih. 2021. Domain-Specific Knowledge Graphs: A Survey. *Journal of Network and Computer Applications*, 185:103076.

Dimitrios Alivanistos, Selene Baez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as Probing: Using Language Models for Knowledge Base Construction. In *LM-KBC@ISWC*, pages 11–34.

Vahan Arsenyan, Spartak Bughdaryan, Fadi Shaya, Kent Wilson Small, and Davit Shahnazaryan. 2024. Large language models for biomedical knowledge graph construction: Information extraction from EMR notes. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 295–317, Bangkok, Thailand. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. SAC-KG: Exploiting large language models as skilled automatic constructors for domain knowledge graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4345–4360, Bangkok, Thailand. Association for Computational Linguistics.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1856–1869, Dubrovnik, Croatia. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. DeepSeek-V3 Technical Report. *Preprint*, arXiv:2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric Xing, and Zhiting Hu. 2023. BertNet: Harvesting knowledge graphs with arbitrary relations from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5000–5015, Toronto, Canada. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2024. GPTKB: Comprehensively

Materializing Factual LLM Knowledge. *Preprint*, arXiv:2411.04920.

Wenyu Huang, Guancheng Zhou, Mirella Lapata, Pavlos Vougiouklis, Sebastien Montella, and Jeff Z. Pan. 2024. Prompting Large Language Models with Knowledge Graphs for Question Answering Involving Long-tail Facts. *Preprint*, arXiv:2405.06524.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large Language Models Struggle to Learn Long-Tail Knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861, Florence, Italy. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The dawn after the dark: An empirical study on factuality hallucination in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10879–10899, Bangkok, Thailand. Association for Computational Linguistics.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.*, 14(1):50–60.

Xiaoze Liu, Feijie Wu, Tianyang Xu, Zhuo Chen, Yichi Zhang, Xiaoqian Wang, and Jing Gao. 2024. Evaluating the Factuality of Large Language Models using Large-Scale Knowledge Graphs. *Preprint*, arXiv:2404.00942.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Linhao Luo, Trang Vu, Dinh Phung, and Reza Haf. 2023. Systematic assessment of factual knowledge in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13272–13286, Singapore. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-tail: How knowledgeable are large language models (LLMs)? A.K.A. will LLMs replace knowledge graphs? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 311–325, Mexico City, Mexico. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *Preprint*, arXiv:2302.13971.

Zeno van Cauter and Nikolay Yakovets. 2024. Ontology-guided knowledge graph construction from maintenance short texts. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 75–84, Bangkok, Thailand. Association for Computational Linguistics.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation. *Preprint*, arXiv:2307.03987.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Somin Wadhwa, Adit Krishnan, Runhui Wang, Byron C Wallace, and Luyang Kong. 2024. Learning from natural language explanations for generalizable entity matching. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6114–6129, Miami, Florida, USA. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Bowen Zhang and Harold Soh. 2024. Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9820–9836, Miami, Florida, USA. Association for Computational Linguistics.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Preprint*, arXiv:2309.01219.

Danna Zheng, Mirella Lapata, and Jeff Z. Pan. 2024. How Reliable are LLMs as Knowledge Bases? Re-thinking Facutality and Consistency. *Preprint*, arXiv:2407.13578.

## A Computing Infrastructure

We run our experiments using a machine with 8 V100 GPUs.

## B Schemas for all Domains

We provide a full schema for a KG about *books* in Figure 3 (the simplified schema was given in Figure 1) and about *landmarks* domain in Figure 4. During schema post-processing, we make sure that the format is correct (two top keys are "concepts" and "relations", all entities have "attributes" etc). We also ensure that relations connect pairs of generated entities and that all relations contain main entity.

## C Prompts for Schema Generation

As described in Section 2.2, our schemas are generated using a two-step prompt-based approach where the first step generates a list of entities and relations among them, while the second step completes the schema by adding attributes to entities. In both steps, we supply a human-crafted schema for a different domain inside a prompt. In this way, model is more successful in outputting the desired structure and content of the schema. Prompts for two steps of schema generation are given in Tables 7 and 8.

## D Prompts for Data Generation

We list prompts for all three steps of data generation in Tables 9, 10 and 11.

## E Prompts for Chain of Verification

We provide prompts for different steps of Chain of Verification in Tables 12, 13, 14 and 15. These prompts are taken from an open-source implementation.[9]

```
{
    "concepts": {
        "book": {
            "attributes": {
                "title": "string",
                "language": "string",
                "publication_date": "string",
                "ISBN": "string",
                "pages": "integer"
            }
        },
        "author": {
            "attributes": {
                "author_name": "string",
                "nationality": "string",
                "birth_date": "string",
                "genre_specialization": "string"
            }
        },
        "publisher": {
            "attributes": {
                "publisher_name": "string",
                "founding_year": "string",
                "location": "string"
            }
        },
        "genre": {
            "attributes": {
                "genre_name": "string",
                "description": "string"
            }
        },
        "award": {
            "attributes": {
                "award_name": "string",
                "type": "string",
                "organizer": "string",
                "venue": "string",
                "award_description": "string"
            }
        },
        "character": {
            "attributes": {
                "character_name": "string",
                "description": "string"
            }
        },
        "series": {
            "attributes": {
                "series_name": "string",
                "description": "string"
            }
        }
    },
    "relations": {
        "author_writes_book": {
            "source": "author",
            "target": "book",
            "name": "writes"
        },
        "book_published_by_publisher": {
            "source": "book",
            "target": "publisher",
            "name": "published_by"
        },
        "book_belongs_to_genre": {
            "source": "book",
            "target": "genre",
            "name": "belongs_to"
        },
        "book_wins_award": {
            "source": "book",
            "target": "award",
            "name": "wins"
        },
        "book_features_character": {
            "source": "book",
            "target": "character",
            "name": "features"
        },
        "book_belongs_to_series": {
            "source": "book",
            "target": "series",
            "name": "belongs_to"
        }
    }
}
```

Figure 3: Full schema for *books* domain, containing 7 entities (`book`, `author`, `publisher`, `genre`, `award`, `character` and `series`) with attributes such as ISBN of a book or author's date of birth, and relations between the `main entity` and all other entities.

[9] https://github.com/ritun16/chain-of-verification

**Prompt for Step 1 of Schema Generation**

INSTRUCTION: You are an assistant for creating knowledge graphs and their schemas. You are provided with an example of schema for knowledge graph about <source_domain> in JSON format, given within triple quotes. This schema contains entities and relations. It does not contain attributes. Following this format, create a schema for the knowledge graph about <target_domain>.
"""
<source_domain_schema>
"""

Table 7: The prompt for the first step of schema generation: This prompt generates a set of entities and relations in JSON format (which is the format of the provided schema for source domain).

**Prompt for Step 2 of Schema Generation**

INSTRUCTION: You are an assistant for creating knowledge graphs and their schemas. You are provided with an example of schema for knowledge graph about <source_domain> in JSON format, given within triple quotes. You are also provided with a partial schema for <target_domain>, given within triple backticks. Following the schema about <source_domain>, modify the schema for <target_domain> by adding attributes to all entities. Output the full JSON schema for <target_domain>.
Schema for <source_domain>:
"""
<source_domain_schema>
"""

Schema for <target_domain>:
```
<target_domain_schema>
```

Table 8: The prompt for the second step of schema generation: This prompt generates a final schema in JSON format. It takes a partial schema obtained in Step 1, and adds attributes to all entities.

**Prompt for Step 1 of Data Generation: Initial Relation Generation**

INSTRUCTION: Generate $N$ different pairs (<entity1>, <entity2>) with the relation "<relation>". Use [SEP] to separate two entities.

**Example**

INSTRUCTION: Generate 20 different pairs (book, genre) with the relation "book_belongs_to_genre". Use [SEP] to separate two entities.

Table 9: The prompt for the first step of data generation: **initial relation generation**.

| **Prompt for Step 2 of Data Generation: Entity Completion** |
|---|
| INSTRUCTION: You are given a JSON template within triple quotes. Your task is to fill this template with information about <entity1> "<identifier1>" and <entity2> "<identifier2>". Ensure that valid information is generated and use "None" for missing or uncertain information. Output the result in JSON format.<br>"""<br><schema><br>""" |

| **Example** |
|---|
| INSTRUCTION: You are given a JSON template within triple quotes. Your task is to fill this template with information about author "J.R.R. Tolkien" and book "The Lord of the Rings". Ensure that valid information is generated and use "None" for missing or uncertain information. Output the result in JSON format.<br>"""<br><schema><br>""" |

Table 10: The prompt for the second step of data generation: **entity completion**. Given two entities and their identifiers, the model fills the schema with all information. The schema used here contains only entities and attributes, while relations are implicit, i.e. the model is expected to generate all entities related to the given pair of entities.

| **Prompt for Step 3 of Data Generation: Entity Generation** |
|---|
| INSTRUCTION: Generate $N$ <entity>s with <input_entity> <input_entity_value>. If there are no matching <entity>s output "None".<br>1. <example> |

| **Example** |
|---|
| INSTRUCTION: Generate 10 books with genre Modernist literature. If there are no matching books output "None".<br>1. Mrs. Dalloway |

Table 11: The prompt for the third step of data generation: **entity generation**. Given two entities and their identifiers, the model generates $N$ identifiers for "main entity" (book in the example).

| **Prompt 1 for CoVe: Generate Question Templates** |
|---|
| INSTRUCTION: Your task is to create verification questions based on the provided question. NO ADDITIONAL TEXT.<br>Example Question: Who are some movie actors born in Boston?<br>Example Verification Question 1: Where was [movie actor] born?<br>Example Verification Question 2: Was [movie actor] born in [Boston]?<br>Explanation: In the above example, the verification questions focused only on the ANSWER_ENTITY (name of the movie actor) and QUESTION_ENTITY (birth place). Similarly, you need to focus on the ANSWER_ENTITY and QUESTION_ENTITY from the actual question and generate verification questions.<br>Actual Question: <original_question><br><br>Verification Question: |

Table 12: This prompt generates templates which will be used to formulate actual verification questions. Given the original query from Step 3. **Entity Generation**, the model generates suitable question templates.

**Prompt 2 for CoVe: Generate Verification Questions**

INSTRUCTION: Your task is to create a series of verification questions based on the given question, the verification question template and baseline response. NO ADDITIONAL TEXT.
Example Question: Who are some movie actors who were born in Boston?
Example Baseline Response: 1. Matt Damon
2. Chris Evans
Example Verification Question Template 1: Was [movie actor] born in Boston?
Example Verification Question Template 2: Where was [movie actor] born?
Example Verification Questions:
1. Was Matt Damon born in Boston?
2. Was Chris Evans born in Boston?
1. Where was Matt Damon born?
2. Where was Chris Evans born?

Actual Question: <original_question>
Baseline Response: <baseline_response>
Verification Question Template: <verification_question_template>

Final Verification Questions:

Table 13: Based on the generated question templates, original question and response, the model is asked to generate verification questions for all entities from the original response.

**Prompt 3 for CoVe: Answer Verification Question**

INSTRUCTION: Answer the following question correctly.

Question: <verification_question>

Answer:

Table 14: Model is asked to answer all verification questions generated in the previous step. Questions are answered independently, one at a time to avoid the interference among the answers.

**Prompt 4 for CoVe: Generate Final (Updated) Answer**

INSTRUCTION: Given the below 'Original Query' and 'Baseline Answer', analyze the 'Verification Questions and Answers' to obtain the final, refined answer. Output the final answer as a numbered list. Include only correct entities. NO ADDITIONAL TEXT.
Original Query: <original_question>
Baseline Answer: <baseline_response>

Verification Questions and Answer Pairs:
<verification_answers>

Final Refined Answer:

Table 15: Based on the original query, initial response, verification questions and answers, the model is asked to update the initial response by keeping only correct entities.

```
{
    "concepts": {
        "landmark": {
            "attributes": {
                "landmark_name": "string",
                "type": "string",
                "year_built": "string",
                "historical_significance": "string"
            }
        },
        "city": {
            "attributes": {
                "city_name": "string",
                "population": "integer"
            }
        },
        "country": {
            "attributes": {
                "country_name": "string",
                "capital": "string",
                "population": "integer",
                "government_type": "string"
            }
        },
        "architect": {
            "attributes": {
                "architect_name": "string",
                "nationality": "string",
                "birth_date": "string",
                "notable_works": "string"
            }
        },
        "style": {
            "attributes": {
                "style_name": "string",
                "description": "string",
                "characteristics": "string"
            }
        },
        "event": {
            "attributes": {
                "event_name": "string",
                "date": "string",
                "type": "string",
                "description": "string"
            }
        },
        "organization": {
            "attributes": {
                "organization_name": "string",
                "type": "string",
                "founding_year": "string",
                "mission": "string"
            }
        }
    },
    "relations": {
        "landmark_located_in_city": {
            "source": "landmark",
            "target": "city",
            "name": "located_in"
        },
        "landmark_located_in_country": {
            "source": "landmark",
            "target": "country",
            "name": "located_in"
        },
        "landmark_designed_by_architect": {
            "source": "landmark",
            "target": "architect",
            "name": "designed_by"
        },
        "landmark_represents_style": {
            "source": "landmark",
            "target": "style",
            "name": "represents"
        },
        "landmark_hosted_event": {
            "source": "landmark",
            "target": "event",
            "name": "hosted"
        },
        "landmark_managed_by_organization": {
            "source": "landmark",
            "target": "organization",
            "name": "managed_by"
        }
    }
}
```

Figure 4: Full schema for *landmarks* domain, containing 7 entities (`landmark`, `city`, `country`, `architect`, `style`, `event` and `organization`), and relations between the `main entity` `landmark` and all other entities.

| **Generated Entity** | **`Wikidata` Item or Property** |
|---|---|
| landmark | architectural structure (Q811979) |
| city | street address (P6375), location (P276) or territory (P131) |
| country | country (P17) |
| architect | architect (P84) |
| style | architectural style (P149) |
| event | - |
| organization | occupant (P466), operator (P137), owner (P127) or maintainer (P126) |

Table 16: `Wikidata` categories used for evaluating KG about *landmarks*. Note that we consider several properties for entities `city` and `organization` to increase the chance of finding the generated entity among them. We also find no matching property for entity `event`, so we exclude it from the evaluation.