Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction

Anonymous ACL submission

Abstract

In this work, we are interested in automated methods for knowledge graph creation (KGC) from input text. Progress on large language models (LLMs) has prompted a series of recent works applying them to KGC, e.g., via zero/few-shot prompting. Despite successes on small domain-specific datasets, these models face difficulties scaling up to text common in many real-world applications. A principal issue is that, in prior methods, the KG schema has to be included in the LLM prompt to generate valid triplets; larger and more complex schema easily exceed the LLMs' context window length. Furthermore, there are scenarios where a fixed pre-defined schema is not available and we would like the method to construct an intrinsically high-quality KG with accurate information and a succinct self-generated schema. To address these problems, we propose a three-phase framework named Extract-Define-Canonicalize (EDC): open information extraction followed by schema definition and post-hoc canonicalization. EDC is flexible in that it can be applied to settings where a pre-defined target schema is available and when it is not; in the latter case, it constructs a schema automatically and applies self-canonicalization. To further improve performance, we introduce a trained component that retrieves schema elements relevant to the input text; this improves the LLMs' extraction performance in a retrieval-augmented generation-like manner. We demonstrate on three KGC benchmarks that EDC is able to extract high-quality triplets without any parameter tuning and with significantly larger schemas compared to prior works.

1 Introduction

Knowledge graphs (KGs) (Ji et al., 2021) are a structured representation of knowledge that organizes interconnected information through graph structures, where entities and relations are represented as nodes and edges. They are broadly

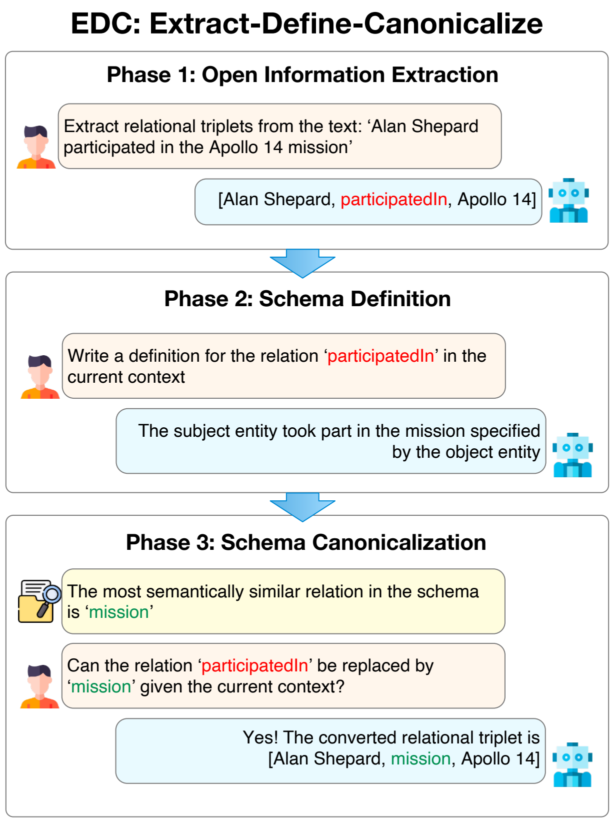

Figure 1: A high-level illustration of Extract-Define-Canonicalize (EDC) for Knowledge Graph Construction.

used in a variety of downstream tasks such as decision-making (Guo et al., 2021; Lan et al., 2020), question-answering (Huang et al., 2019; Yasunaga et al., 2021), and recommendation (Guo et al., 2020; Wang et al., 2019). However, knowledge graph construction (KGC) is inherently challenging: the task requires competence in understanding syntax and semantics to generate a consistent, concise, and meaningful knowledge graph. As such, KGC predominantly relies on intensive human labor (Ye et al., 2022).

Recent attempts to automate KGC (Zhong et al., 2023; Ye et al., 2022) have employed large language models (LLMs) in view of their remarkable natural language understanding and generation capabilities. LLM-based KGC methods employ various innovative prompt-based techniques, such as multi-turn conversation (Wei et al., 2023) and code generation (Bi et al., 2024), to generate entity-relation triplets that represent the knowledge graph. However, these methods are currently limited to small and domain-specific scenarios — to ensure the validity of generated triplets, schema information (e.g., possible entity and relation types) has to be included in the prompt. Complex datasets (e.g., Wikipedia) typically require **large schemas that exceed the context window length** or can be ignored by the LLMs (Wadhwa et al., 2023). Furthermore, **pre-defined schemas are not always available** — the users might not have pre-determined or fixed intentions about what information is of interest in advance but still would like to extract intrinsically high-quality KGs in a more flexible manner. It is unclear how existing methods will work in such situations.

To address these problems, we propose **Extract-Define-Canonicalize** (**EDC**), a structured approach for KGC: the key idea is to decompose KGC into three primary phases corresponding to three subtasks (Fig. 1):

1. Open Information Extraction: extract a list of entity-relation triplets from the input text freely.

2. Schema Definition: generate a definition for each component of the schema, e.g. entity type and relation type, induced by triplets obtained in the extraction phase.

3. Schema Canonicalization: use the schema definitions to standardize the triplets such that semantically-equivalent entities/relations types have the same noun/relation phrase.

Each phase exploits the strengths of LLMs: the Extract subtask leverages recent findings that LLMs are effective open information extractors (Li et al., 2023; Han et al., 2023) — they can extract semantically correct and meaningful triplets. However, the resulting triplets typically contain redundant and ambiguous information, e.g., multiple semantically equivalent relation phrases such as 'profession', 'job', and 'occupation' (Kamp et al., 2023; Putri et al., 2019; Vashishth et al., 2018).

Phases 2 and 3 (Define and Canonicalize) standardize the triplets to make them useful for downstream tasks. We designed EDC to be flexible: it can either discover triplets consistent with a pre-existing schema of potentially large size (**Target Alignment**) or *self-generate* a schema (**Self Canonicalization**). To achieve this, we use LLMs to define the schema components by exploiting their explanation generation capabilities — LLMs can justify their extractions via explanations that are agreeable to human experts (Li et al., 2023). The definitions are used to find the closest entity/relation type candidates (via a vector similarity search) that the LLM can then reference to canonicalize a component. In the case there is no equivalent counterpart in the existing schema, we can choose to add it to enrich the schema.

To further improve performance, the three steps above can be followed by an additional **Refinement** phase: we repeat EDC but provide the previously extracted triplets and a relevant part of the schema in the prompt during the initial extraction. We propose a trained **Schema Retriever** that retrieves schema components relevant to the input text (akin to retrieval-augmented generation (Lewis et al., 2020)), which we find improves the generated triplets.

Experiments on three KGC datasets in both Target Alignment and Self Canonicalization settings show that EDC is able to extract higher-quality KGs compared to state-of-the-art methods through both automatic and manual evaluation. Furthermore, the use of the Schema Retriever is shown to significantly and consistently improve EDC's performance.

In summary, the paper makes the following contributions:

- EDC, a flexible and performant LLM-based framework for knowledge graph construction that is able to extract high-quality KGs with schema of large size or without any pre-defined schema.

- Schema Retriever, a trained model to extract schema components relevant to input text in the same vein as information retrieval.

- Empirical evidence that demonstrate the effectiveness of EDC and the Schema Retriever.

2 Background

In this section, we provide relevant background on knowledge graph construction (KGC), open information extraction (OIE), and canonicalization.

Knowledge Graph Construction. Traditional methods typically addressed KGC using "pipelines", comprising subtasks like entity discovery (Žukov-Gregorič et al., 2018; Martins et al., 2019), entity typing (Choi et al., 2018; Onoe and Durrett, 2020), and relation classification (Zeng et al., 2014, 2015). Thanks to advances in pre-trained generative language models (e.g., T5 (Raffel et al., 2020) and BERT(Lewis et al., 2019)), more recent works instead frame KGC as a sequence-to-sequence problem and generate relational triplets in an end-to-end manner by fine-tuning these moderately-sized language models (Ye et al., 2022). The success of large language models (LLMs) has pushed this paradigm further: current methods directly prompt the LLMs to generate triplets in a zero/few-shot manner. For example, ChatIE (Wei et al., 2023) extracts triplets by framing the task as a multi-turn question-answering problem and CodeKGC (Bi et al., 2024) approaches the task as a code generation problem. As previously mentioned, these models face difficulties scaling up to general text common in many real-world applications as the KG schema has to be included in the LLM prompt. Our EDC framework circumvents this problem by using post-hoc canonicalization (and without requiring fine-tuning of the base LLMs).

Open Information Extraction and Canonicalization. Standard (closed) information extraction requires the output triplets to follow a pre-defined schema, e.g. a list of relation or entity types to be extracted from. In contrast, open information extraction (OIE) does not have such a requirement. OIE has a long history and we refer readers who want comprehensive coverage to the excellent surveys (Liu et al., 2022; Zhou et al., 2022; Kamp et al., 2023). Recent studies have found LLMs to exhibit excellent performance on OIE tasks (Li et al., 2023). However, the relational triplets extracted from OIE systems are not canonicalized, e.g. multiple semantically equivalent relations can coexist without being unified to a canonical form, causing redundancy and ambiguity in the induced open knowledge graph. An extra canonicalization step is required to standardize the triplets to make the KGs useful for downstream applications.

Canonicalization methods differ depending on whether a target schema is available. In case a target schema is present, the task is sometimes referred to as "alignment" (Putri et al., 2019). For example, (Putri et al., 2019) uses WordNet (Miller, 1995) as side information to obtain definitions for the OIE-extracted relation phrases and a Siamese network to compare an OIE relation definition and a pre-defined relation in the target schema. In case no target schema is available, state-of-the-art methods are commonly based on clustering (Vashishth et al., 2018; Dash et al., 2020). CESI (Vashishth et al., 2018) creates embeddings for the OIE relations using side information from external sources like PPDB (Ganitkevitch et al., 2013) and WordNet. However, clustering-based methods are prone to over-generalization (Kamp et al., 2023; Putri et al., 2019), e.g., CESI may put "is brother of," "is son of," "is main villain of," and "was professor of" into the same relation cluster.

Compared to the existing canonicalization methods, EDC is more general; it works whether a target schema is provided or not. Instead of using static external sources like WordNet, EDC utilizes contextual and semantically-rich side information generated by LLMs. Furthermore, by allowing the LLMs to verify if a transformation can be performed (instead of solely relying on the embedding similarity), EDC alleviates the over-generalization issue faced by previous methods.

3 Method: EDC for KGC

This section outlines our primary contribution: an approach to constructing knowledge graphs that leverages LLMs in a structured manner. We first detail the EDC framework followed by a description of refinement (**EDC+R**). Given input text, our goal is to extract relational triplets in a canonical form such that the resulting KGs will have minimal ambiguity and redundancy. When there is a pre-defined target schema, all generated triplets should conform to it. In the scenario where there is not one, the system should dynamically create one and canonicalize the triplets with respect to it.

3.1 EDC: Extract-Define-Canonicalize

At a high level, EDC decomposes KGC into three connected subtasks. To ground our discussion, we will use a specific input text example: "*Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew*" and walk through each of the phases:

Phase 1: Open Information Extraction: we first leverage Large Language Models (LLMs) for open information extraction. Through few-shot prompting, LLMs identify and extract relational triplets ([Subject, Relation, Object]) from input texts, independent of any specific schema. Using our example above, the prompt is:

> Given a piece of text, extract relational triplets in the form of [Subject, Relation, Object] from it.
> Here are some examples:
> Example 1:
> Text: The 17068.8 millimeter long ALCO RS-3 has a diesel-electric transmission.
> Triplets: [['ALCO RS-3', 'powerType', 'Diesel-electric transmission'], ['ALCO RS-3', 'length', '17068.8 (millimetres)']] ...
> Now please extract triplets from the following text: Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew.

The resultant triplets (in this case, ['Alan Shepard', 'bornOn', 'Nov 18, 1923'], ['Alan Shepard', 'participatedIn', 'Apollo 14']) form an *open KG*, which is forwarded to subsequent phases.

Phase 2: Schema Definition: Next, we prompt the LLMs to provide a natural language definition for each component of the schema induced by the open KG:

> Given a piece of text and a list of relational triplets extracted from it, write a definition for each relation present.
> Example 1:
> Text: The 17068.8 millimeter long ALCO RS-3 has a diesel-electric transmission.
> Triplets: [['ALCO RS-3', 'powerType', 'Diesel-electric transmission'], ['ALCO RS-3', 'length', '17068.8 (millimetres)']]
> Definitions:
> powerType: The subject entity uses the type of power or energy source specified by the object entity.
> ...
> Now write a definition for each relation present in the triplets extracted from the following text:
> Text: Alan Shepard was an American who was born on Nov 18, 1923 in New Hampshire, was selected by NASA in 1959, was a member of the Apollo 14 crew and died in California
> Triplets: [['Alan Shepard', 'bornOn', 'Nov 18, 1923'], ['Alan Shepard', 'participatedIn', 'Apollo 14']]

This example prompt results in the definitions for (bornOn: The subject entity was born on the date specified by the object entity.) and (participatedIn: The subject entity took part in the event or mission specified by the object entity.), which are then passed to the next stage as *side information* used for canonicalization.

Phase 3: Schema Canonicalization: The third phase aims to refine the open KG into a canonical form, eliminating redundancies and ambiguities. We start by vectorizing the definitions of each schema component using a sentence transformer to create embeddings. Canonicalization then proceeds in one of two ways, depending on the availability of a target schema:

- Target Alignment: With an existing target schema, we identify the most closely related components within the target schema for each element, considering them for canonicalization. To prevent issues of over-generalization, LLMs assess the feasibility of each potential transformation. If a transformation is deemed unreasonable, indicating no semantic equivalent in the target schema, the component, and its related triplets are excluded.
- Self Canonicalization: Absent a target schema, the goal is to consolidate semantically similar schema components, standardizing them to a singular representation to streamline the KG. Starting with an empty canonical schema, we examine the open KG triplets, searching for potential consolidation candidates through vector similarity and LLM verification. Unlike target alignment, components deemed non-transformable are added to the canonical schema, thereby expanding it.

Using our example, the prompt is:

> Given a piece of text, a relational triplet extracted from it, and the definition of the relation in it, choose the most appropriate relation to replace it in this context if there is any.
> Text: Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew.
> Triplets: ['Alan Shepard', 'participatedIn', 'Apollo 14']
> Definition of 'participatedIn': The subject entity took part in the event or mission specified by the object entity.
> Choices:
> A. 'mission': The subject entity participated in the event or operation specified by the object entity.
> B. 'season': The subject entity participated in the season of a series specified by the object entity.
> ...
> F. None of the above

Note that the choices above are obtained by using vector similarity search. After the LLM makes its choice, the relations are transformed to yield:

['Alan Shepard', 'birthDate', 'Nov 18, 1923'], ['Alan Shepard', 'mission', 'Apollo 14'], which forms our canonicalized KG.

3.2 EDC+R: iteratively refine EDC with Schema Retriever

The refinement process leverages the data generated by EDC to enhance the quality of the extracted triplets. Inspired by retrieval-augmented generation and prior work (Bi et al., 2024), we construct a "hint" for the extraction phase (details in Appendix A.4), which comprises two main elements:

- Candidate Entities: The entities extracted by EDC from the previous iteration, and entities extracted from the text using the LLM;
- Candidate Relations: The relations extracted by EDC from the previous cycle and relations retrieved from the pre-defined/canonicalized schema by using a trained Schema Retriever.

The inclusion of entities and relations from both the LLM and the schema retriever provides a richer pool of candidates for the LLM, which addresses issues where the absence of entities or relations impairs the LLM's effectiveness. By merging the entities and relations extracted in earlier phases with new findings from entity extraction and schema retrieval, the hint serves to aid the OIE by bootstrapping from the previous round.

To scale EDC to large schemas, we employ a trained Schema Retriever which allows us to efficiently search schemas. The Schema Retriever works in a similar fashion to information retrieval methods based on vector spaces (Ganguly et al., 2015; Lewis et al., 2020); it projects the schema components and the input text to a vector space such that cosine similarity captures the relevance between the two, i.e., how likely a schema component to be present in the input text. Note that in our setting, the similarity space is different from the standard sentence embedding models where cosine similarity in the vector space captures semantic equivalence. Our Schema Retriever is a fine-tuned variant of the sentence embedding model E5-mistral-7b-instruct (Wang et al., 2023). We follow the original training methodology detailed in the paper, which involves utilizing pairs of text and their corresponding defined relations. For details, please refer to the Appendix A.3. For a given positive text-relation pair (t^+, r^+), we employ an instruction template on t^+ to generate a new text t^+_{inst} = "Instruct: retrieve relations that are present in the given text $\backslash n$ Query: $\{t^+\}$".

We then finetune the embedding model to distinguish between the correct relation associated with a given text and other non-relevant relations using the InfoNCE loss.

Back to our example, refinement with the schema retriever adds the following relation to the previous set: ['Alan Shepard', 'selectedByNasa', '1959']. The relation 'selectedByNasa' is rather obscure but was specified in the target schema.

4 Experiments

In this section, we describe experiments designed to evaluate the performance of EDC and EDC+R. Briefly, our results demonstrate that EDC significantly outperforms the state-of-the-art methods in both Target Alignment and Self Canonicalization settings. Refinement further improves EDC. Source code for EDC and to replicate our experiments are available in the supplementary materials, with full tables in the Appendix C.

4.1 Experimental Setup

Datasets. We evaluate EDC using three KGC datasets:

- WebNLG (Ferreira et al., 2020): We use the test split from the semantic parsing task of WebNLG+2020 (v3.0). It contains 1165 pairs of text and triplets. The schema derived from these reference triplets encompasses 159 unique relation types.
- REBEL (Cabot and Navigli, 2021): The original test partition of REBEL comprises 105,516 entries. To manage costs, we select a random sample of 1000 text-triplet pairs. This subset induces a schema with 200 distinct relation types.
- Wiki-NRE (Distiawan et al., 2019): From Wiki-NRE's test split (29,619 entries), we sample 1000 text-triplet pairs, resulting in a schema with 45 unique relation types.

These datasets were chosen over alternatives like ADE (Gurulingappa et al., 2012) (1 relation type), SciERC (Luan et al., 2018) (7 relation types), and CoNLL04 (Roth and Yih, 2004) (4 relation types) used to evaluate previous LLM-based methods (Bi et al., 2024; Wadhwa et al., 2023) used in prior LLM-based studies, due to their richer variety of relation types. This diversity better mimics real-world complexities. In our experiments, we focus on extracting relations as the only schema component available across all datasets. Relations, being a foundational element of KGs, are prioritized over other components like entity or event types. However, note that EDC can be readily extended to other schema components.

EDC Models. EDC contains multiple modules that are powered by LLMs. Since the OIE module is the key upstream module that determines the semantic content captured in the KG, we tested different LLMs of different sizes including GPT-4 (Achiam et al., 2023), GPT-3.5-turbo (Brown et al., 2020), and Mistral-7b (Jiang et al., 2023). Mistral-7b was deployed on a local workstation, whereas the GPT models were accessed via the OpenAI API. For the framework's remaining components which required prompting, we used GPT-3.5-turbo. In the canonicalization phase, the E5-Mistral-7b model was utilized for vector similarity searches without modifications.

4.1.1 Evaluation Criteria and Baselines

We evaluate our methods differently under Target Alignment (when a schema is provided) and Self Canonicalization (no schema) due to the *inherently different objectives*: the former aims to recover the ground-truth annotated triplets consistent with the target schema while the latter is to extract semantically correct and meaningful triplets that induce a succinct and non-redundant KG without a pre-defined target to compare against. For the datasets above, the preivous LLM-based KGC methods (ChatIE and CodeKGC) could not be used due to the schema size. Although EDC is not intended for small domain-specific datasets, we include the results on SciERC and CoNLL04 in the Appendix E for the comprehensiveness of the evaluation.

Target Alignment. We compare EDC and EDC+R against the specialized trained models for each of the datasets:

- **REGEN** (Dognin et al., 2021) is the SOTA model for WebNLG. It is a sequence-to-sequence model that leverages pre-trained T5 (Raffel et al., 2020) and Reinforcement Learning (RL) for bidirectional text-to-graph and graph-to-text generation.

- **GenIE** (Josifoski et al., 2022), a sequence-to-sequence model that leverages pre-trained BART (Lewis et al., 2019) and a constrained generation strategy to constrain the output triplets to be consistent with the pre-defined schema. GenIE is the state-of-the-art model for REBEL and Wiki-NRE.

Following previous work (Dognin et al., 2021; Melnyk et al., 2022), we use the WEBNLG evaluation script (Ferreira et al., 2020) which computes the Precision, Recall, and F1 scores for the output triplets against the ground truth in a token-based manner. Metrics based on Named Entity Evaluation were used to measure the Precision, Recall, and F1 score in three different ways.

- *Exact:* Requires a complete match between the candidate and reference triple, disregarding the type (subject, relation, object).

- *Partial:* Allows for at least a partial match between the candidate and reference triple, disregarding the type.

- *Strict:* Demands an exact match between the candidate and reference triplet, including the element types.

Self Canonicalization. For evaluating self-canonicalization performance, comparisons are made with:

- **Baseline Open KG**, which is the initial open KG output from the OIE (Open Information Extraction) phase. This serves as a reference point to illustrate the changes in precision and schema conciseness resulting from the canonicalization process.

- **CESI** (Vashishth et al., 2018), recognized as a leading clustering-based approach for open KG canonicalization. By applying CESI to the open KG, we aim to contrast its performance against canonicalization by EDC.

Given that canonicalized triplets may use relations phrased differently from the reference triplets or entirely out-of-schema relations, a token-based evaluation becomes unsuitable. Thus, we resort to manual evaluation, focusing on three key aspects that reflect the intrinsic quality of an extracted KG:

- *Precision:* The canonicalized triplets remain correct and meaningful with respect to the text compared to the OIE triplets.

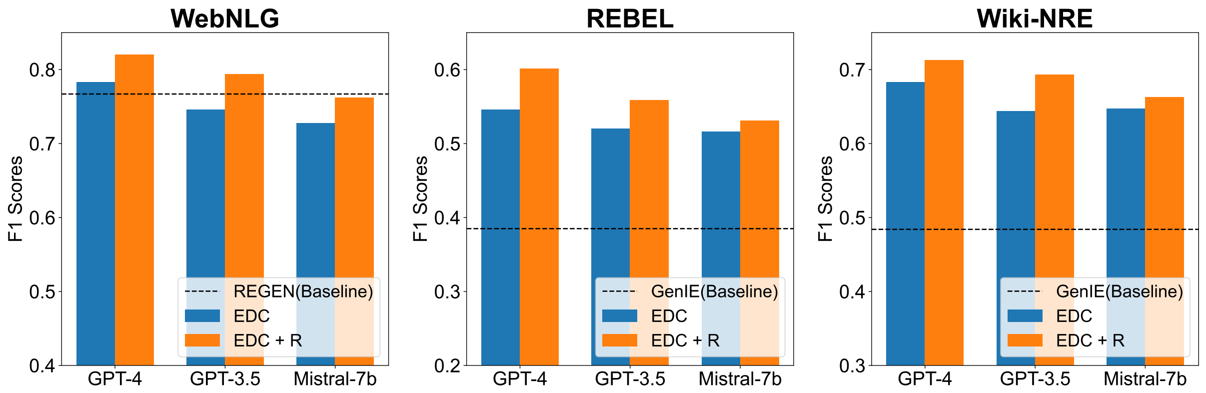

Figure 2: Performance of EDC and EDC+R on WebNLG, REBEL, and Wiki-NRE datasets against the respective baselines in the Target Alignment setting (F1 scores with 'Partial' criteria). EDC+R only performs 1 iteration of refinement as we found the improvement diminishes significantly afterward.

- *Conciseness:* The schema's brevity is measured by the number of relations types.

- *Redundancy:* We employ a redundancy score — the average cosine similarity among each canonicalized relation and its nearest counterpart — where low scores indicate that the schema's relations are semantically distinct.

4.2 Results and Analysis

In the following, we focus on conveying our main findings and results. For full results and tables, please refer to the Appendix.

4.2.1 Target Alignment

The bar charts in Figure 2 summarize the Partial F1 scores obtained by EDC and EDC+R on all three datasets with different LLMs for OIE compared against the respective baselines. **EDC demonstrates performance that is superior to or on par with the state-of-the-art baselines for all evaluated datasets**. Comparing the LLMs, GPT-4 emerges as the top performer, with Mistral-7b and GPT-3.5-turbo exhibiting comparable results. The disparity between our methods and the baselines is more pronounced on the REBEL and Wiki-NRE datasets; this is primarily due to the GenIE's constrained generation approach, which falls short in extracting triplets that include literals, such as numbers and dates.

Refinement (EDC+R) consistently and significantly enhances performance. Post-refinement, the difference in performance between GPT-3.5-turbo and Mistral-7b is larger, suggesting Mistral-7b's was not as able to leverage the provided hints. Nevertheless, a single refinement iteration with the hint improved performance for all the tested LLMs.

From the scores, it appears that EDC performance is significantly better on WebNLG compared to REBEL and Wiki-NRE. However, we observed that EDC was penalized despite producing valid triplets on the latter datasets. A reason for this is that the reference triplets in these datasets are non-exhaustive. For example, given the text in the REBEL dataset, '*Romany Love is a 1931 British musical film directed by Fred Paul and starring Esmond Knight, Florence McHugh and Roy Travers.*', EDC extracts: ['Romany Love', 'cast member', 'Esmond Knight'], ['Romany Love', 'cast member', 'Florence McHugh'], ['Romany Love', 'cast member', 'Roy Travers'], which are all semantically correct, but only the first triplet is present in the reference set. The datasets also contain reference triplets based on information extraneous to the text, e.g., '*Daniel is an Ethiopian footballer, who currently plays for Hawassa City S.C.*' has a corresponding reference triplet ['Hawassa City S.C.', 'country', 'Ethiopia'].

These issues can be attributed to the distinct methodologies employed in the creation of these datasets. For WebNLG, annotators were asked to compose text solely from the triplets. Thus, the text and the triplets have a direct correspondence, and the text typically does not include information other than what is apparent from the triplets. In contrast, REBEL and Wiki-NRE are created by aligning text and triplets using distant supervision (Smirnova and Cudré-Mauroux, 2018). This method can lead to less straightforward triplets to extract and incomplete reference sets, which can lead to pessimistic evaluations for methods such as EDC that produce correct triplets not in the dataset (Han et al., 2023; Wadhwa et al., 2023). On average, EDC extracts 1 more triplet per sentence compared to the reference set on REBEL and Wiki-NRE, compared to WebNLG where EDC extracts a similar number of triplets to the reference.

Table 1: Ablation study results (F1 scores with all criteria) on schema retriever, the LLM used for OIE is GPT-3.5-turbo. S.R. stands for Schema Retriever.

Dataset	Method	Partial	Strict	Exact
WebNLG	EDC+R	**0.794**	**0.753**	**0.772**
	EDC+R w/o S.R.	0.752	0.701	0.721
	EDC	0.746	0.688	0.713
REBEL	EDC+R	**0.559**	**0.516**	**0.529**
	EDC+R w/o S.R.	0.517	0.466	0.482
	EDC	0.506	0.449	0.473
Wiki-NRE	EDC+R	**0.693**	**0.685**	**0.657**
	EDC+R w/o S.R.	0.653	0.645	0.641
	EDC	0.647	0.638	0.640

Ablation study on schema retriever. To evaluate the impact of the relations provided by the schema retriever during refinement, we conducted an ablation study with GPT-3.5-turbo by removing the relations retrieved using the schema retriever from the hint. The results in Table 1 show that **ablating the Schema Retriever leads to a notable decline in performance**. Qualitatively, we find that the schema retriever helps to find relevant relations that are challenging for the LLMs to identify during the OIE stage. For example, given the text *'The University of Burgundy in Dijon has 16,800 undergraduate students'*, the LLMs extract ['University of Burgundy', 'location', 'Dijon'] during OIE. Although semantically correct, this relation overlooks the more specific relation present in the target schema, namely 'campus', for denoting university's location. The schema retriever successfully identifies this finer relation, enabling the LLMs to adjust their extraction to ['University of Burgundy', 'campus', 'Dijon']. This experiment highlights the schema retriever's value in facilitating the extraction of precise and contextually appropriate relations.

4.2.2 Self Canonicalization

Here, we focus on evaluating EDC's self-canonicalization performance (utilizing GPT-3.5-turbo for OIE). We omit refinement in Self Canonicalization setting as it has already been studied above and in subsequent iterations, the self-constructed canonicalized schema becomes the target schema. Following prior work (Wadhwa et al., 2023; Kolluru et al., 2020), we conducted a targeted human evaluation of knowledge graphs. This evaluation involved two independent annotators assessing the reasonableness of triplet extractions from given text without prior knowledge of the system's details. We observed a high inter-annotator agreement score of 0.94.

Table 2: Performance of EDC in the Self Canonicalization setting (human-evaluated precision and schema metrics). The best result for each dataset and metric is bolded. Prec. stands for precision, No. Rel. stands for the number of relations and Red. stands for redundancy score

Dataset	Method	Prec.(↑)	No. Rel.(↓)	Red.(↓)
WebNLG	EDC	**0.956**	**200**	**0.833**
	CESI	0.724	280	0.893
	Open KG	0.982	529	0.927
REBEL	EDC	**0.867**	**225**	**0.831**
	CESI	0.504	307	0.854
	Open KG	0.903	667	0.895
Wiki-NRE	EDC	**0.898**	**106**	**0.833**
	CESI	0.753	114	0.849
	Open KG	0.909	204	0.881

The evaluation results and schema metrics are summarized in Table 2. These findings reveal that while the open KG generated by the OIE stage contains semantically valid triplets (which affirms the previous findings that LLMs are competent open information extractors (Li et al., 2023)), it suffers from a significant degree of redundancy within the resultant schema. **EDC accurately canonicalizes the open KG and yields a schema that is both more concise and less redundant compared to CESI**. EDC avoids CESI's tendency toward overgeneralization — in line with prior work (Putri et al., 2019), we observed CESI inappropriately clusters diverse relations such as 'place of death', 'place of birth', 'date of death', 'date of birth', and 'cause of death' into a single 'date of death' category.

5 Conclusion

In this work, we presented EDC, an LLM-based three-phase framework that addresses the problem of KGC by open information extraction followed by post-hoc canonicalization. Experiments show that EDC and EDC+R are able to extract better KGs than specialized trained models when a target schema is available and dynamically create a schema when none is provided. The scalability and versatility of EDC opens up many opportunities for applications: it allows us to automatically extract high-quality KGs from general text using large schemas like Wikidata (Vrandečić and Krötzsch, 2014) and even enrich these schemas with newly discovered relations.

6 Limitations

There are several limitations that we would like to address in future works. First, we only considered schema canonicalization within the scope of this paper, it is of great interest to incorporate an entity de-duplication mechanism in the future to reduce the redundancy in the constructed KGs further, e.g. via coreference resolution (Sukthanker et al., 2020). Moreover, EDC's components can be further improved to boost the performance, e.g. the schema retriever may benefit from training on more diverse and higher-quality data. Finally, due to time and resource constraints, we only tested different LLMs for OIE while all the other modules of EDC rely on GPT-3.5-turbo, it will be beneficial to test the smaller open-source models' performance on the other tasks as well.

7 Ethical Considerations

Artifact usage. The datasets we used in the paper are only leveraged for research purposes and we strictly follow the corresponding licenses (e.g. WebNLG uses cc-by-nc-sa-4.0). It is to be noted that, due to the nature of the task, the datasets may inherently contain information about individuals (especially celebrities). We project to make the software and code for this paper publicly available under the MIT license.

Human annotators. The two annotators (1 male and 1 female) are recruited university students. The annotators are compensated fairly and given abundant and flexible time to complete the tasks. The collection protocol is determined exempt by an IRB board.

Potential Risks. It needs to be noted that the use of current LLMs may bring risks such as hallucinations (Xu et al., 2024) and privacy issues (Yao et al., 2024).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2020. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. arXiv preprint arXiv:2010.12688.

Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. 2024. Codekgc: Code language model for generative knowledge graph construction. ACM Transactions on Asian and Low-Resource Language Information Processing, 23(3):1–16.

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

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. Rebel: Relation extraction by end-to-end language generation. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2370–2381.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. arXiv preprint arXiv:1807.04905.

Sarthak Dash, Gaetano Rossiello, Nandana Mihindukulasooriya, Sugato Bagchi, and Alfio Gliozzo. 2020. Open knowledge graphs canonicalization using variational autoencoders. arXiv preprint arXiv:2012.04780.

Bayu Distiawan, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 229–240.

Pierre L Dognin, Inkit Padhi, Igor Melnyk, and Payel Das. 2021. Regen: Reinforcement learning for text and knowledge base generation using pretrained language models. arXiv preprint arXiv:2108.12472.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris Van Der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+).

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, pages 795–798.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies, pages 758–764.

Liang Guo, Fu Yan, Yuqian Lu, Ming Zhou, and Tao Yang. 2021. An automatic machining process decision-making system based on knowledge graph. International journal of computer integrated manufacturing, 34(12):1348–1369.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering, 34(8):3549–3568.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. Journal of biomedical informatics, 45(5):885–892.

Ridong Han, Tao Peng, Chaohao Yang, Benyou Wang, Lu Liu, and Xiang Wan. 2023. Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors. arXiv preprint arXiv:2305.14450.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In Proceedings of the twelfth ACM international conference on web search and data mining, pages 105–113.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE transactions on neural networks and learning systems, 33(2):494–514.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. arXiv preprint arXiv:2310.06825.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. GenIE: Generative information extraction. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4626–4643, Seattle, United States. Association for Computational Linguistics.

Serafina Kamp, Morteza Fayazi, Zineb Benameur-El, Shuyan Yu, and Ronald Dreslinski. 2023. Open information extraction: A review of baseline techniques, approaches, and applications. arXiv preprint arXiv:2310.11644.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Soumen Chakrabarti, et al. 2020. Openie6: Iterative grid labeling and coordination analysis for open information extraction. arXiv preprint arXiv:2010.03147.

Luong Thi Hong Lan, Tran Manh Tuan, Tran Thi Ngan, Nguyen Long Giang, Vo Truong Nhu Ngoc, Pham Van Hai, et al. 2020. A new complex fuzzy inference system with fuzzy knowledge graph and extensions in decision making. Ieee Access, 8:164899–164921.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

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

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. arXiv preprint arXiv:2304.11633.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173.

Pai Liu, Wenyang Gao, Wenjie Dong, Songfang Huang, and Yue Zhang. 2022. Open information extraction from 2007 to 2022–a survey. arXiv preprint arXiv:2208.08690.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. arXiv preprint arXiv:1808.09602.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2019. Joint learning of named entity recognition and entity linking. arXiv preprint arXiv:1907.08243.

Igor Melnyk, Pierre Dognin, and Payel Das. 2022. Knowledge graph generation from text. arXiv preprint arXiv:2211.10511.

George A Miller. 1995. Wordnet: a lexical database for english. Communications of the ACM, 38(11):39–41.

Yasumasa Onoe and Greg Durrett. 2020. Fine-grained entity typing for domain independent entity linking. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8576–8583.

Rifki Afina Putri, Giwon Hong, and Sung-Hyon Myaeng. 2019. Aligning open ie relations and kb relations using a siamese network based on word embedding. In Proceedings of the 13th International Conference on Computational Semantics-Long Papers, pages 142–153.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140):1–67.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Proceedings of the eighth conference on computational natural language learning (CoNLL-2004) at HLT-NAACL 2004, pages 1–8.

Alisa Smirnova and Philippe Cudré-Mauroux. 2018. Relation extraction using distant supervision: A survey. ACM Computing Surveys (CSUR), 51(5):1–35.

Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. 2020. Anaphora and coreference resolution: A review. Information Fusion, 59:139–162.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In Proceedings of the 2018 World Wide Web Conference, pages 1317–1327.

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

Somin Wadhwa, Silvio Amir, and Byron C Wallace. 2023. Revisiting relation extraction in the era of large language models. In Proceedings of the conference. Association for Computational Linguistics. Meeting, volume 2023, page 15566. NIH Public Access.

Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In The world wide web conference, pages 3307–3313.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. arXiv preprint arXiv:2401.00368.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. arXiv preprint arXiv:2302.10205.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. arXiv preprint arXiv:2401.11817.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. High-Confidence Computing, page 100211.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. arXiv preprint arXiv:2104.06378.

Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. 2022. Generative knowledge graph construction: A review. arXiv preprint arXiv:2210.12714.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In Proceedings of the 2015 conference on empirical methods in natural language processing, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers, pages 2335–2344.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. ACM Computing Surveys, 56(4):1–62.

Shaowen Zhou, Bowen Yu, Aixin Sun, Cheng Long, Jingyang Li, Haiyang Yu, Jian Sun, and Yongbin Li. 2022. A survey on neural open information extraction: Current status and future directions. arXiv preprint arXiv:2205.11725.

Andrej Žukov-Gregorič, Yoram Bachrach, and Sam Coope. 2018. Named entity recognition with parallel recurrent neural networks. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 69–74.

A Implementation Details

A.1 Models and Infrastructures Details

We use two OpenAI models, GPT-3.5-turbo and GPT-4 (sizes currently unknown), and an open-source model, Mistral-7b (7 billion parameters). The training and inference of open-source models are done with a single machine with an AMD EPYC 7543P 32-Core Processor and 252GB of RAM, equipped with 4 NVIDIA RTX A6000 (48GB) GPUs. We accessed GPT-3.5-turbo and GPT-4 via the OpenAI API.

A.2 Prompting-related hyperparameters

We use few-shot prompting for all modules of EDC, we empirically choose 6-shot examples from the respective datasets. For the MCQ used in the Schema Canonicalization phase, we retrieve top-5 semantically similar relations from the schema as candidates. For refinement, the schema retriever retrieves top-10 most relevant relations from the schema as candidate relations. These hyperparameters are empirically chosen to balance performance and inference costs.

A.3 Schema Retriever Training

We follow the original training methodology detailed in the original paper (Wang et al., 2023), which involves utilizing pairs of text and their corresponding defined relations. For a given positive text-relation pair (t^+, r^+), we employ an instruction template on t^+ to generate a new text $t^+_{inst} =$ "Instruct: retrieve relations that are present in the given text $\backslash n$ Query: $\{t^+\}$".

We then finetune the embedding model to distinguish between the correct relation associated with a given text and other non-relevant relations using the InfoNCE loss,

$$\min \mathcal{L} = -\log \frac{\phi(t^+_{inst}, r^+)}{\phi(t^+_{inst}, r^+) + \sum_{n_i \in N} \phi(t^+_{inst}, n_i)}$$

Here, N denotes the set of negative samples, and ϕ represents the cosine similarity function. Please see the appendix for additional training details.

For training, we synthesized a dataset of text-relation pairs using the TEKGEN dataset (Agarwal et al., 2020), a large-scale text-triplets dataset created by aligning Wikidata triplets to Wikipedia text. The training dataset comprised 37,500 pairs, evenly divided between positive and negative samples. We adopted an online open-source implementation and hyperparameter configurations for training.

The performance of the fine-tuned schema retriever was assessed on the test splits of WebNLG, REBEL, and Wiki-NRE datasets. The recall@10 scores on these datasets were 0.823, 0.663, and 0.818, respectively, indicating the effectiveness of the retriever across different knowledge graph contexts.

A.4 Details of Refinement Hint

The refinement hint consists of candidate entities and candidate relations. This section details the obtainment of them and how they are used to improve the OIE performance. We will carry on using the example used in Section 3: "*Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew*" and the triplets extracted by EDC in the first iteration are ['Alan Shepard', 'birthDate', 'Nov 18, 1923'], ['Alan Shepard', 'mission', 'Apollo 14'].

A.4.1 Obtaining Candidate Entities

The candidate entities come from two sources:

- Entities extracted by EDC from the previous iteration, i.e. ['Alan Shepard', 'Nov 18, 1923', 'Apollo 14']
- Entities extracted from the text by prompting the LLM to perform an entity extraction task, similar to the triplet extraction task.

> Given a piece of text, extract a list of entities from it.
> Here are some examples:
> Example 1:
> Text: The 17068.8 millimeter long ALCO RS-3 has a diesel-electric transmission.
> Entities: ['ALCO RS-3', 'Diesel-electric transmission', '17068.8 (millimetres)']
> ...
> Now please extract entities from the following text: Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew.

and the resultant entities are ['Alan Shepard', 'Nov 18, 1923', 'NASA', '1959', 'Apollo 14']

The entities are then merged together as the candidate entities.

A.4.2 Obtaining Candidate Relations

The candidate relations also come from two sources:

- Relations extracted by EDC from the previous iteration, i.e. ['birthDate', 'mission']
- Relations extracted by the schema retriever, by calculating the relevance score between the input text and the relations in the schema. The top 5 retrieved relations in this case are ['birthDate', , 'selectedByNasa', 'mission', 'draftPick', 'occupation'].

The relations and their corresponding definitions are then merged together as the candidate relations. It is to be noted that, similar to other RAG-based methods, there is a chance that the retriever may retrieve irrelevant information. In this case, the relation definitions can come in handy as it provides more information for the LLMs to decide whether the relation is a valid one with respect to the text or not.

A.4.3 Usage of Hint for Refined OIE

The refinement hint is then included in the prompt appropriately to instruct the LLMs to consider (but is not limited to) the candidate entities and candidate relations:

> Given a piece of text, extract relational triplets in the form of [Subject, Relation, Object] from it.
> Here are some examples:
> Example 1:
> Text: The 17068.8 millimeter long ALCO RS-3 has a diesel-electric transmission.
> Entities: ['ALCO RS-3', 'Diesel-electric transmission', '17068.8 (millimetres)']
> Triplets: [['ALCO RS-3', 'powerType', 'Diesel-electric transmission'], ['ALCO RS-3', 'length', '17068.8 (millimetres)']]
> ...
> Now please extract triplets from the following text: Alan Shepard was born on Nov 18, 1923 and selected by NASA in 1959. He was a member of the Apollo 14 crew. Entities: ['Alan Shepard', 'Nov 18, 1923', 'NASA', '1959', 'Apollo 14']
> Here are some potential relations and their descriptions you may look out for during extraction:
> 1. birthDate: The subject entity was born on the date specified by the object entity.
> 2. mission: The subject entity participated in the event or operation specified by the object entity.
> 3. selectedByNasa: The subject entity was selected by NASA in the year specified by the object entity.
> ...

The extracted triplets by the refined OIE are:['Alan Shepard', 'birthDate', 'Nov 18, 1923'], ['Alan Shepard', 'mission', 'Apollo 14'], ['Alan Shepard', 'selectedByNasa', '1959']. It successfully recovers the subtle and fine-grained relation 'selectedByNasa' that would have been missed without using the hint. Also, the semantically rich descriptions help the LLM avoid excessively extracting noisy relations retrieved by the schema retriever.

We found it important to include the entities from both sources, i.e. extractions from the last round and discovered by a separate module (entity extraction or schema retriever). The significance of the schema retriever is already shown in the ablation study in Sec 4.2.1.

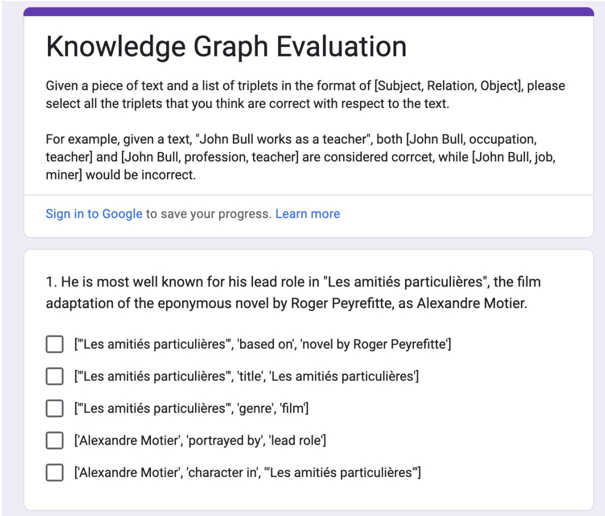

Figure 3: An example screenshot of the questionnaire including the instructions given to the annotators.

B Annotation Instruction

An example screenshot is provided in Figure 3 to illustrate the format of questionnaires and instructions the annotators are given. The purpose of collection of the data was communicated to the annotators verbally.

C Detailed Results of Target Alignment

C.1 Complete Results

The complete results of EDC and EDC+R on WebNLG, REBEL and Wiki-NRE are summarized in Table 3, Table 4 and Table 5 respectively. EDC performs better than or comparable to state-of-the-art baseline models in terms of all metrics (Precision, Recall, and F1) in all criteria (Partial, Strict, and Exact) and EDC+R is able to consistently improve upon this in all aspects as well. These results more comprehensively demonstrate the performance of EDC and EDC+R.

C.2 Effect of More Refinement Iterations

Table 6 shows the result of an extra iteration of refinement with EDC on all datasets. Although further refinement improves the results in a stable manner, we observe diminishing returns and hence, only report one iteration in the main results.

C.3 Ablation Study on Last-Round Extractions

Table 7 shows the result of ablating the relations and entities from the last round's extractions from the refinement hint. It shows the importance of incorporating them, i.e., the importance of performing the refinement in an iterative manner. Merging the two sources led to better coverage of the entities and relations in the text, resulting in better and more stable improvement KGC.

C.4 Discussion on KGC Dataset Annotations

As stated in Section 4.2, we observe that EDC is penalized by the scorer on Rebel and Wiki-NRE datasets due to incomplete annotations. This echoes the previous finding in (Wadhwa et al., 2023; Han et al., 2023) that LLMs can often extract correct results that are missing in the annotations, which results in overly pessimistic evaluations. As shown by Table 8, EDC tends to extract significantly more triplets compared to the reference annotations and the baseline GenIE. Furthermore, as shown from the manual evaluation in Table 2, many of these triplets are indeed meaningful and correct with respect to the input text. Regardless, despite the automatic evaluation result on EDC being overly pessimistic, it still exceeds the baseline by a large margin and the actual performance may be even larger considering the difference in the number of triplets extracted.

D Additional Experiments on Novel Fictional Dataset

Since the tested datasets are already from several years ago and the training set of the LLMs used are not known, there is a risk the LLMs have already been trained on the datasets. To address this concern, we create a novel small-scale dataset (50 entries) of fictional entities and information, e.g. “Evergreen University was where Emily Johnson received her degree in Biology” and annotated them using the schema of Wiki-NRE. As illustrated by the results in Table 9, EDC and EDC+R still obtain very strong performance superior to the baseline GenIE model, showing that the performance cannot be trivially explained by data leakage.

E Comparison against previous LLM-based approaches

Although this is not the intended use scenario for EDC, we include these experimental results for a more comprehensive evaluation to compare against existing LLM-based methods. We conduct experiments on three datasets, CoNLL04 (4 relation types) (Roth and Yih, 2004), SciERC (7 relation types) (Luan et al., 2018) and our sub-sampled version of Wiki-NRE (45 relation types), which is the only dataset we use in our main experiments that can fit in the context window. To ensure comparison fairness, we use GPT-3.5-turbo for all the compared methods.

As illustrated by the results in Table 10, when the relation number is small (CONLL and SciERC), EDC alone may not be superior to the baseline methods due to not incorporating the schema in the prompt. However, through refinement, EDC+R is able to achieve significantly better results. This may be attributed to the usage of the semantically rich relation descriptions in the refinement step. Specifically, it helps correct two types of errors that may occur during extraction: 1. the Definition step helps disambiguate homonyms, e.g., the word "follows" has two different meanings for "John follows Taoism" v.s. "John follows Mary". EDC changes the "follows" in "John follows Taoism" to "adheres to". 2. Using the relation definitions, we find the Refinement step corrects head-tail relation errors, e.g., for a relation "father", it is unclear if the subject or the object is the father, and the definition prevents inconsistent use. This error-correcting effect was not possible in previous methods.

When tested on Wiki-NRE, which has a moderately-sized schema, EDC already significantly outperforms the baseline methods, potentially due to the confusion of the LLMs when dealing with long context (Liu et al., 2024). Furthermore, we observe that ChatIE and CodeKGC may still output out-of-schema relation words although the entire schema is provided in the prompt, echoing the previous findings (Wadhwa et al., 2023).

Table 3: Complete results of EDC and EDC+R on WebNLG dataset against the baseline REGEN (Precision, Recall, F1 with 'Partial', 'Strict' and 'Exact' criteria). EDC+R only performs 1 iteration of refinement. The best results are bolded.

		Partial			Strict			Exact		
Method	LLM for OIE	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
EDC	GPT-4	**0.776**	**0.796**	**0.783**	**0.729**	**0.741**	**0.733**	**0.751**	**0.765**	**0.756**
	GPT-3.5	0.739	0.760	0.746	0.684	0.697	0.688	0.708	0.722	0.713
	Mistral-7b	0.723	0.739	0.728	0.668	0.679	0.672	0.692	0.703	0.696
EDC+R	GPT-4	**0.814**	**0.831**	**0.820**	**0.782**	**0.794**	**0.786**	**0.796**	**0.808**	**0.800**
	GPT-3.5	0.788	0.806	0.794	0.749	0.761	0.753	0.768	0.781	0.772
	Mistral-7b	0.756	0.775	0.762	0.716	0.727	0.720	0.735	0.747	0.739
Baseline	REGEN	0.755	0.788	0.767	0.713	0.735	0.720	0.714	0.738	0.723

Table 4: Complete results of EDC and EDC+R on REBEL dataset against the baseline REGEN (Precision, Recall, F1 with 'Partial', 'Strict', and 'Exact' criteria). EDC+R only performs 1 iteration of refinement. The best results are bolded.

		Partial			Strict			Exact		
Method	LLM for OIE	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
EDC	GPT-4	**0.543**	**0.552**	**0.546**	**0.498**	**0.503**	**0.500**	**0.511**	**0.517**	**0.514**
	GPT-3.5	0.503	0.512	0.506	0.448	0.453	0.449	0.471	0.476	0.473
	Mistral-7b	0.512	0.523	0.516	0.450	0.457	0.453	0.481	0.488	0.483
EDC+R	GPT-4	**0.599**	**0.606**	**0.601**	**0.557**	**0.561**	**0.559**	**0.572**	**0.576**	**0.574**
	GPT-3.5	0.556	0.565	0.559	0.513	0.519	0.516	0.527	0.533	0.529
	Mistral-7b	0.525	0.550	0.531	0.461	0.462	0.462	0.506	0.511	0.505
Baseline	GENIE	0.381	0.391	0.385	0.353	0.361	0.356	0.362	0.369	0.364

Table 5: Complete results of EDC and EDC+R on Wiki-NRE dataset against the baseline REGEN (Precision, Recall, F1 with 'Partial', 'Strict', and 'Exact' criteria). EDC+R only performs 1 iteration of refinement. The best results are bolded.

		Partial			Strict			Exact		
Method	LLM for OIE	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
EDC	GPT-4	**0.682**	**0.686**	**0.683**	**0.675**	**0.679**	**0.677**	**0.676**	**0.680**	**0.678**
	GPT-3.5	0.645	0.651	0.647	0.636	0.640	0.638	0.638	0.643	0.640
	Mistral-7b	0.644	0.650	0.647	0.636	0.640	0.637	0.637	0.641	0.639
EDC+R	GPT-4	**0.712**	**0.715**	**0.713**	**0.708**	**0.710**	**0.709**	**0.708**	**0.711**	**0.709**
	GPT-3.5	0.691	0.696	0.693	0.684	0.688	0.685	0.685	0.689	0.687
	Mistral-7b	0.661	0.667	0.663	0.647	0.652	0.649	0.656	0.661	0.658
Baseline	GENIE	0.482	0.486	0.484	0.462	0.464	0.463	0.477	0.479	0.478

Table 6: Results (F1 scores with all criteria) of further iterative refinement, the LLM used for OIE is GPT-3.5-turbo. EDC+2xR is EDC with 2 iterations of refinement.

	WebNLG			REBEL			Wiki-NRE		
Method	Partial	Strict	Exact	Partial	Strict	Exact	Partial	Strict	Exact
EDC+2xR	0.797	0.761	0.775	0.564	0.521	0.535	0.697	0.689	0.660
EDC+R	0.794	0.753	0.772	0.559	0.516	0.529	0.693	0.685	0.657
EDC	0.746	0.688	0.713	0.506	0.449	0.473	0.644	0.634	0.637

Table 7: Results (F1 scores with all criteria) of ablating the entities and relations extracted from the last round from the refinement hint, the LLM used for OIE is GPT-3.5-turbo. EDC+R-lastround is EDC with refinement but entities and relations extracted from the last round are removed from the refinement hint.

	WebNLG			REBEL			Wiki-NRE		
Method	Partial	Strict	Exact	Partial	Strict	Exact	Partial	Strict	Exact
EDC+R	0.794	0.753	0.772	0.559	0.516	0.529	0.693	0.685	0.657
EDC+R-lastround	0.748	0.698	0.720	0.534	0.485	0.505	0.634	0.622	0.625
EDC	0.746	0.688	0.713	0.506	0.449	0.473	0.644	0.634	0.637

Table 8: The average number of triplets extracted per sentence on all three datasets. The baseline model for WebNLG is REGEN while the baseline for Rebel and Wiki-NRE is GENIE. Numbers in the brackets are the difference from the reference annotations.

LLM for OIE	WebNLG	REBEL	Wiki-NRE
GPT-4	3.47(+0.04)	5.11(+1.11)	3.49(+0.63)
GPT-3.5	3.44(+0.01)	5.01(+1.01)	3.49(+0.63)
Mistral7b	3.45+(0.02)	4.68(+0.68)	3.75(+0.89)
Baseline	-	2.20(-1.80)	3.08(+0.22)
Reference	3.43	4.00	2.86

Table 9: Complete results of EDC and EDC+R on the novel fictional dataset against the baseline GenIE (Precision, Recall, F1 with 'Partial', 'Strict' and 'Exact' criteria). EDC+R only performs 1 iteration of refinement. The best results are bolded. The LLM used for OIE is GPT-3.5-turbo.

	Partial			Strict			Exact		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
EDC	0.731	0.771	0.751	0.687	0.704	0.691	0.702	0.720	0.707
EDC+R	0.761	0.782	0.767	0.733	0.750	0.738	0.733	0.750	0.738
GenIE	0.521	0.547	0.530	0.426	0.443	0.432	0.467	0.483	0.472

Table 10: Complete results of EDC, EDC+R on CONLL, SciERC and Wiki-NRE datasets against the previous LLM-based approaches, CodeKGC and ChatIE. The LLMs used here are GPT-3.5-turbo to ensure comparison fairness. The best results are bolded.

		Partial			Strict			Exact		
Dataset	Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
CONLL	EDC	0.536	0.552	0.543	0.481	0.491	0.485	0.503	0.515	0.509
	EDC+R	**0.580**	**0.593**	**0.585**	**0.514**	**0.522**	**0.517**	**0.549**	**0.558**	**0.552**
	CodeKGC	0.542	0.55	0.545	0.503	0.506	0.504	0.542	0.546	0.543
	ChatIE	0.463	0.477	0.468	0.360	0.366	0.363	0.418	0.427	0.421
SciERC	EDC	0.389	0.408	0.395	0.288	0.301	0.292	0.352	0.365	0.357
	EDC+R	**0.447**	**0.461**	**0.451**	**0.340**	**0.349**	**0.343**	**0.406**	**0.416**	**0.410**
	CodeKGC	0.389	0.398	0.392	0.277	0.283	0.279	0.346	0.353	0.349
	ChatIE	0.351	0.367	0.357	0.212	0.221	0.215	0.294	0.302	0.297
Wiki-NRE	EDC	0.645	0.651	0.647	0.636	0.640	0.638	0.638	0.643	0.640
	EDC+R	**0.691**	**0.696**	**0.693**	**0.684**	**0.688**	**0.685**	**0.685**	**0.689**	**0.687**
	CodeKGC	0.611	0.614	0.612	0.605	0.607	0.606	0.607	0.609	0.608
	ChatIE	0.569	0.574	0.571	0.541	0.545	0.543	0.553	0.557	0.555